HCLSoftware

HCL DevOps Loop 2026.08 (2.0.2.1) User Documentation

Release date: 14 August 2026

Special notice

Before using this information and the product it supports, read the information in Notices on page mmmccxcii.

Contents

Chapter 1. Release Notes..... 5
What's New..... 5
Defect fixes..... 19
Known issues..... 20
Chapter 2. System Requirements..... 25
Hardware..... 26
Operating systems and containers..... 26
Host prerequisites..... 27
Supported software..... 29
Chapter 3. Getting Started..... 30
Overview of DevOps Loop..... 30
User interface..... 30
Accessibility features..... 36
Chapter 4. Administration..... 37
Configuration of external databases for the capabilities in DevOps Loop..... 37
Configuring an external database for Plan and Control..... 37
Configuring an external database for Deploy..... 39
Configuring an external database for Build..... 41
Configuring an external database for Test..... 42
Configuring an external database for Measure and Release..... 43
Configuring external access to the Plan PostgreSQL database..... 44
Installation of DevOps Loop..... 45
Installing DevOps Loop on IBM Cloud Kubernetes Service (IKS)..... 46
Installing DevOps Loop on Kubernetes Service (K8S)..... 49
Installation of DevOps Loop on Google Kubernetes Engine (GKE)..... 56
Installing DevOps Loop on a RHEL system for a demo setup..... 59
Installation of DevOps Loop on Amazon Web Services (AWS)..... 62
Installation of DevOps Loop in an air-gapped environment..... 69
Managing DevOps Loop features after installation..... 76
License management and user administration..... 77
DevOps Loop licensing information..... 78
Enabling the social sign-up and social login for DevOps Loop..... 80
Revoking a license..... 81
User access and administration using Keycloak..... 81
About user roles and access permissions..... 83
Backup and restoration of DevOps Loop..... 84
Backing up DevOps Loop data..... 85
Restoring DevOps Loop data..... 86
Private CA and self-signed certificate support..... 87
Configuring trusted certificates in DevOps Loop..... 88
Upgrading DevOps Loop..... 90
Integrations in DevOps Loop..... 93
AI provider integration for Loop Genie - Tech Preview..... 94
Harbor integration overview..... 104
DevOps Loop MCP server..... 111
Getting started with the DevOps Loop MCP server..... 112
DevOps Loop MCP server configuration in VS Code..... 127
Connecting Claude to the DevOps Loop MCP server..... 130
Configuring an external MCP server in a teamspace..... 130
Teamspace management..... 132
Creating a teamspace..... 132
Adding or inviting members to a teamspace..... 134
Joining a teamspace as an invited member..... 135
Removing a member from a teamspace..... 135
Uninstalling DevOps Loop..... 136
Uninstalling DevOps Loop from IBM Cloud Kubernetes Service (IKS)..... 136
Uninstalling DevOps Loop from Kubernetes Service (K8S)..... 137
Chapter 5. Working with Loops..... 139
Loop management..... 139
Creating a loop..... 140
Capabilities and resources configured during loop creation..... 142
Viewing loop details..... 145
Adding members to a loop..... 145
Removing a member from a loop..... 146
Disabling a loop..... 147
Enabling a loop..... 147
Loop Segments overview..... 148
Learning Loop overview..... 151
Creating a Loop with preloaded example data..... 152
Dashboards and insights..... 154
Viewing dashboards..... 155
Editing dashboards..... 155
AI-powered search..... 156
Chapter 6. Capabilities of DevOps Loop..... 158
Plan..... 158
Code implementation with AI..... 158

Code....159
Code overview....159
Accessing dev containers....161
Dev containers....162
Viewing logs....165
Working with dev containers....169
Configuring dev containers....172
Code IDE user interface....180
Extensions in dev containers....181
File management features....182
Access applications via automatic port forwarding....184
File system in dev containers....184
Loading extensions from local .vsix files....186
Configuring extension registry....187
Git authentication for dev containers....191
Code Genie....199
Creating Plan work items from Code....217
Installing GitHub Copilot in dev containers....223
Control....224
AI code reviewing with Code Genie....224
Trivy scanning with Code Genie....228
Branching strategy with a protected main branch....233
Managing Git authentication for developers....233
Build....234
System requirements....234
Overview of Build....235
Tutorials....245
Administering....299
Creating and Managing Projects....351
Build pipelines....374
Creating Jobs and Processes....375
Managing Project Templates....383
Monitoring and Reporting....393
Extending Product Functions....401
Test....699
Introduction to DevOps Test Hub....699
Capabilities....703
User tasks....703
Member access and permissions....704
Compatibility and supported assets....704
Accessibility features....708
Tutorials....708
Administrator Guide....730
Test Author Guide....760
Test Execution Specialist Guide....1025
Test Manager Guide....1713
Troubleshooting....1787
Release....1798
Introduction to releases and pipelines....1798
Home page....1798
Release....
Administering....1799
Releasing Software....1831
Extending Product Functions....1874
How to....1897
Glossary....1898
Deploy....1904
Getting Started....1904
Installing an external agent for Deploy in DevOps Loop....2055
Integrating....2057
Administering....2057
Managing Security....2149
Modeling Software Deployment....2244
Deploying....2525
Reporting....2547
Extending Product Function....2556
Troubleshooting and Support....2997
Measure....3016
Introduction to releases and pipelines....3016
Home page....3016
Measure....
Administering....3017
Managing Value Streams....3049
Developing Data Insights....3163
Releasing Software....3203
Extending Product Functions....3246
How to....3269
Glossary....3270
Chapter 7. Loop Genie overview - Tech Preview....3277
Loop Genie workspace....3278
Interacting with Loop Genie....3281
Loop Genie tasks....3282
Running a predefined task in Loop Genie....3284
Creating or modifying a task in Loop Genie....3285
Prompt references for Loop Genie....3286
Chapter 8. Troubleshooting....3290
Security Considerations....mmmccxci
Notices....mmmccxcii
Index....3295

Chapter 1. Release notes for DevOps Loop 2026.08 (2.0.2.1)

This document includes information about the features introduced, a product overview, and installation instructions, as well as contact information of HCL Customer Support for this version of HCL DevOps Loop.

Product description

DevOps Loop is a cloud-based continuous integration platform. DevOps Loop is built on modern, cloud-native technologies that enables product teams to plan, code, test, build and deploy the applications efficiently. DevOps Loop provides a holistic view of the progress in the DevOps cycle, allowing teams to track development, monitor workflows, and manage releases effectively. For more information, see Overview of DevOps Loop on page 30.

What's new

You can find information about the features introduced, enhancements, or other changes in DevOps Loop. See What's New on page 5.

Product download and installation

If you have purchased the licenses to use the product, you can download the entitled software packages from HCL Harbor container registry.

For instructions about installing the product software, see Installation of DevOps Loop on page 45.

Known issues

For the list of known issues that are identified in this version and the previously published known issues that are still applicable, see Known issues on page 20.

Contacting HCL support

- For technical assistance, contact HCL Customer Support.
- Before you contact HCL support, you must gather the background information that you might need to describe your problem. When you describe a problem to an HCL Support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:
    - What software versions were you running when the problem occurred?
    - Do you have logs, traces, or messages that are related to the problem?
    - Can you reproduce the problem? If so, what steps do you take to reproduce it?
    - Is there a workaround for the problem? If so, be prepared to describe the workaround.

What's New

You can find information about the features introduced, enhancements, or other changes made in HCL DevOps Loop.

What's New in DevOps Loop 2026.08 (2.0.2.1)

Release date: 14 August 2026

Ther are no new features in this release of DevOps Loop.

What's New in DevOps Loop 2026.06 (2.0.2)

Release date: 09 July 2026

The following table lists the new features, enhancements, or other changes made in DevOps Loop:

Feature title	Description
Azure OpenAI integration for Loop Genie	You can now configure Azure OpenAI as AI provider in Loop Genie, that enables AI-powered workflows through Microsoft Azure services. See Configuring AZURE OpenAI integration on page 98.
External MCP server support in Loop Genie	You can now connect Loop Genie to external MCP servers that enable integration with third-party tools and services through the Model Context Protocol. See Configuring an external MCP server in a teamspace on page 130.
Harbor integration in DevOps Loop	You can configure Harbor as an integrated capability during DevOps Loop installation to provide container image storage, vulnerability scanning, and image management capabilities. See Harbor integration overview on page 104.
Updated user interface	You can experience a refreshed user interface across DevOps Loop with updated page designs that provide a more consistent, modern, and accessible user experience.
Demo data for segments in Learning Loop	You can now create a complete sample DevOps environment with preloaded data across the Plan, Control, Build, Deploy, and Test capabilities. The Learning Loop includes frontend and backend segments with ready-to-use repositories, build pipelines, and deployments that demonstrate end-to-end DevOps workflows. This environment enables hands-on practice without impacting production systems. See Learning Loop overview on page 151.
Enhanced Loop Genie - Tech Preview	Loop Genie now uses an enhanced ReAct execution model that dynamically determines the next action required to satisfy your request, evaluates results after each step, and invokes additional tools as needed, while displaying full tool activity (names, parameters, execution status, and results) inline for complete transparency. See Loop Genie overview - Tech Preview on page 3277.
Plan	
Code Genie support	The Code Genie feature provides an end-to-end integration within the Loop environment that allows you to generate source code directly from a work item. See Code implementation with AI on page 158.
Control	

Feature title	Description
Code Genie support	Control repositories created for Loops are now automatically configured to enable AI-assisted code reviews through Code Genie and the Copilot dev container. You can now use interactive code suggestions across your team during code reviews. See AI code reviewing with Code Genie on page 224.
Deploy	
Azure OpenAI support for Deployment Genie	Added support for Azure OpenAI LLM integration in Deployment Genie for AI-assisted failure analysis. See Configuring the AI settings on page 2117.
Locking snapshot in process step	Added a new built-in step for locking snapshots in generic, application, and component processes to ensure deployment audit requirements. See Creating snapshots on page 2383.
Exporting the draft process	Added the ability to export draft component processes in JSON and YAML formats for review workflows. See Exporting specific component processes on page 2300.
Amazon S3 archiving	Added native support for archiving component versions to Amazon S3 buckets with configurable policies and IAM authentication. See Archiving version artifacts to Amazon S3 on page 2122.
Deployment reports enhancements	The deployment reports are enhanced as follows: • Support for selecting multiple applications and environments simultaneously. • Added approver information. • Added **Snapshot** and **Description** of the deployment request process columns. See Deployment reports on page 2547.
Plugin page enhancements	The Plugin page is enhanced as follows: • The **View Documentation** option is added to the plug-in pages for direct access to plug-in documentation. • The **Installed plugins** page now highlights plug-ins with newer versions available. See Installing plug-ins from the Web UI on page 2560.

Feature title	Description
Message of the Day formatting	Added CommonMark markdown formatting support (bold, links, lists, headers, code blocks) to the **Message of the Day** system settings. See System settings on page 2086.
Enhanced MCP tools	Added the following MCP tools: • **GetDeploymentFailedSteps**: Retrieves information on failed deployment steps. • **GetStepOutput**: Accesses step execution logs and output. • **SearchStepOutputRegex**: Searches step output using RE2-protected regex patterns. • **SearchStepOutputHangs**: Identifies long intervals between log output lines. See Using the Deploy MCP tools on page 2994.
Dependent generic process exclusion	Added an option to exclude dependent generic processes from the import operation. See Importing and exporting generic processes on page 2452.
Additional truststore for server	Added support for an additional server truststore that is used alongside the JVM default truststore. It helps in storing keys specific to webhooks, email, and other TLS clients. See Keystores configuration on page 2164.
Test	
AI-powered test generation with Test Genie in Test Hub	You can now use Test Genie to automatically generate test assets within Test. By providing natural language instructions and connecting to a supported AI provider, you can accelerate your test creation process. While test steps are being generated, you can view AI interactions and session logs on the progress page. After the test generation is complete, you can view the generation metrics. See Test creation with Test Genie on page 780, Creating a Web UI test with Test Genie on page 781, and Modifying a Web UI test created by using Test Genie on page 783.
Support for the Anthropic AI provider	You can now add and manage the Anthropic AI provider on the **Cloud Credentials** page in Test. You can then use Anthropic to generate tests by using Test Genie, or to populate schema generator configurations when auto-assigning generators and configurations to each field in a data definition. See Management of cloud credentials on page 1115, Test creation with Test Genie on page 780, and Auto-assigning generators and AI-provided data configurations to data definitions in the Schema View on page 1009.
Personal agents for local test execution and debugging	You can now download, install, and start a personal agent directly from the **Test Editor** page. A personal agent is a local agent that provides a dedicated, isolated runtime

Feature title	Description
	environment to run Web UI tests and suites directly on your local machine so that you can visually observe UI interactions in real time and identify the exact point of test failure. See Management of the personal agent on page 1095, Installing and starting the personal agent on page 1096, and Stopping or unregistering the personal agent on page 1100.
MCP support for authentication	MCP clients can now dynamically register with the authorization server (Keycloak) without requiring an offline token to be passed through the authentication header. When you connect to an MCP server endpoint in Test to access server-side tools directly from MCP clients, such as VS Code, you must provide your credentials to authenticate the connection. See Configuration of an MCP server endpoint on page 756 and Configuring an MCP server endpoint in VS Code on page 757.
Support for local repository creation and asset management	You can now create standalone local repositories without a remote repository reference. These local repositories operate the same way as the remote repositories, and they can optionally be connected to an empty remote repository at a later point for synchronization. See Adding a repository to a team space on page 1031.
Support for running server test assets and server suite assets on device clouds	You can now run server test assets and server suite assets on Android or iOS devices that are available through supported device clouds, such as BrowserStack Cloud, LambdaTest Cloud, and pCloudy Cloud. See Running server test assets by using a cloud credential on page 1326 and Configuring a run of a Web UI test to run by using a cloud credential on page 1354.
Customization of environments with profile overrides	You can now configure profile overrides in custom environments so that you can easily change the target profile without changing your test script. When you run a test or a suite with a profile override, the default profile, such as a specific browser or emulated device, is substituted with a different target profile, such as a cloud-native device or an alternative browser. See Test environments on page 869, Creating an environment on page 870, and Modifying an environment on page 872.
Device emulation support on Safari for Web UI tests	You can now record and play back Web UI tests on the Safari browser in the device emulation mode. You can use the Phone (X-Large) and Tablet (X-Large) options that are available for Safari on Mac. See Web UI test creation on page 769.
Support for downloading the agent from the project's Agents and Intercepts page	You can now download DevOps Test Runtime from the **Agents and Intercepts** page after navigating to your project from the initial team space. See Installing and registering DevOps Test Runtime agents on page 1087.
Support for recording and playback of jQuery UI controls	You can now record and play back tests for supported jQuery UI controls. The Smartshot view includes a **Filter by jQuery** option that helps you identify supported jQuery UI controls when editing tests. See Editing a Web UI test on page 793.
Support for conditional flow control actions in Web UI tests	You can now use conditional flow control actions such as **If, While, and Do-while** directly in Web UI test steps. These actions support smartshots and use the same

Feature title	Description
	decision logic as verification points, so that you can perform steps based on the presence or state of a UI widget. The **Pause** step is renamed to **Wait** and now supports both **UI** and **General** modes. See Adding an If step on page 816, Adding a While step on page 818, Adding a Do-While step on page 821, and Adding a Wait step on page 802.
Support for moving or deleting tests with dependent assets	You can now move or delete tests along with their dependent assets. Assets that are referenced exclusively by a test are processed together with the test, while shared assets remain unaffected. References to shared assets are updated automatically to prevent any impact on the test run. See Test navigator panel on page 761.
Support for multi-select actions in Test Navigator	You can now select multiple assets in the Test Navigator and perform actions, such as editing, moving, and deleting, on all selected assets, which makes asset management more efficient. See Test navigator panel on page 761.
Easy navigation in Test Editor to access the failed and error steps in the suites	When you play back test suites in **Test Editor**, the three color-coded boxes display the counts of passed steps, failed steps, and errors. You can now view the step details of the first failed step or the step with an error by clicking the respective color-coded box. See Running a suite sequentially on page 834 and Running a suite concurrently on page 835.
Support for viewing individual test results in Test Editor	You can now select and view individual test results directly within **Test Editor**. Result information is displayed in **Test Editor**, and step-level results are highlighted when the selected result matches the current version of the test. See Running a Web UI test on page 785.
Support for viewing configuration details in functional reports and test logs	You can now view configuration details directly in the functional report and test log. Configuration nodes for tests and modules display the initial variable values and dataset information to provide better visibility into the configuration used during test runs. Variable information is displayed in a more streamlined format to improve report readability and analysis. See Test logs on page 1523 and Viewing a Functional Report on page 1511.
Support for generating negative test data from data definitions	You can now generate negative test data from data definitions to validate error handling and input validation scenarios. You can configure the type and amount of negative data that is generated, preview the generated data, and create datasets that combine invalid values, optionally, combined with valid data to support more effective, comprehensive testing. See Generating negative test data for a data definition on page 1015.
Build	
UI Modernization	The user interface has been improved for better navigation and usability.

Feature title	Description
Support for Repository-based configuration files	You can now store Job as Code (JAC) configuration files, such as `<config_file>.json` or `<config_file>.yaml`, anywhere within your source code repository. When configuring a Job as Code step, you can specify the path to the configuration file relative to the repository root. During execution, the plugin retrieves the file from the branch that triggered the build and dynamically generates the build workflow. See Creating Job templates using the Job as Code plugin on page 378.
Measure	
Added Data retention policy	Product administrators can now configure data retention policy to archive or permanently delete historical data for the resource types, such as versions, builds, or issues. You can provide the retention period across VSM, team, or application scopes to optimize database performance, reduce UI clutter, and simplify compliance management.
Enhanced User Interface	The user interface is enhanced for better navigation and usability.
Enhanced MCP token exchange time	The Model Context Protocol (MCP) workflow is enhanced by adding a cache layer. This helps make MCP integration more responsive and reliable.
Added support for Jira, Jenkins, and GitHub	The Analytics dashboards can now display the data for Jira, Jenkins, and GitHub.
Code	
Updated user interface	The interface is refined to reflect a more intuitive user experience, featuring improved navigation and dark theme support.
Code Genie: AI-assisted development and PR reviews	You can now use the AI providers, including but not limited to GitHub Copilot and IBM Bob, to automate pull request workflows and Plan work item implementation with AI-powered code generation, security scanning, and AI-agent code review.
Enhanced extension registry support	You can now configure private extension registries for secure, administrator-controlled extension management with centralized configuration and on-premises hosting support.

You can find information about the features introduced, enhancements, or other changes made in the previous versions of DevOps Loop in the following sections.

- What's New in DevOps Loop 2026.03 (2.0.1) on page 12
- What's New in DevOps Loop 2025.12 (2.0.0) on page 15
- What's New in DevOps Loop 2025.09 (1.0.3) on page 16
- What's new in DevOps Loop 2025.06 (1.0.2) on page 18
- What's new in DevOps Loop 2025.03 (1.0.1) on page 19

What's New in DevOps Loop 2026.03 (2.0.1)

Release date: 13 April 2026

The following table lists the new features, enhancements, or other changes made in DevOps Loop:

Feature title	Description
AWS platform support	You can now install DevOps Loop on AWS. See Installation of DevOps Loop on Amazon Web Services (AWS) on page 62.
Support for loop segments	You can manage multiple microservices within a single loop by creating independent segments with dedicated resources and integrations. See Loop Segments overview on page 148.
Enhanced Loop Genie - AI Assistant	You can now use predefined tasks from the **Tasks** tab to automate workflows and perform common DevOps operations. You can also review or reuse past interactions from the **History** tab within a unified workspace. See Loop Genie overview - Tech Preview on page 3277.
New loop dashboards	You can now see three new dashboards available for your loop - Delivery dashboard, Quality and Security dashboard, and Productivity dashboard. See Dashboards and insights on page 154.
Dashboard enhancements	You can now view the Issue flow rate visualization, which replaces Issue Created vs. Issue Closed and includes total outstanding open issues, eliminating the need for a separate Total open issues chart. You can also view Lead time and Cycle time by calculated using the median (in days) for accurate insights. See Dashboards and insights on page 154.
Plan	
Support for the Plan MCP server	The Model Context Protocol (MCP) server is integrated to connect with AI-powered development environments (such as GitHub Copilot in VS Code), which enables management of DevOps Plan through natural language prompts by using MCP tools, helping in streamlining developer workflows. See Plan MCP endpoint and tools on page 115.
Control	
Support for the Control MCP server	DevOps Control enables AI-driven development by integrating AI agents through the MCP server. Through this integration, you can perform DevOps operations, including repository (create, fork, search) and code management (create/update/delete files, retrieve contents, manage branches), and collaboration workflows (creating, approving, and merging pull requests) by using natural language commands, with support for AI IDEs such as Visual Studio Code. See Control MCP endpoint and tools on page 119.

Feature title	Description
Support for Trivy security scan	DevOps Control repositories created in a loop are now automatically configured with Trivy security scanning, which runs on every pull request before code reaches the main branch. This feature automatically checks for vulnerabilities and misconfigurations, so that teams can enforce severity-based merge protection.
Git credential manager	After any Loop upgrade, you should remove cached credentials from the previous use of Git Credential Manager. See Managing Git authentication for developers.
Git credential manager update	You must remove cached credentials from previous use of Git Credential Manager after upgrading Loop to ensure proper authentication. See Managing Git authentication for developers.
Code	
Support for Log Viewer	You can now use Log Viewer in Code to view, filter, and download logs from dev containers and teamspaces for easier troubleshooting.
Support for Extension Registry - Tech Preview	The Extension Registry is a tech preview feature for configuring local extension registries, so that you can control and publish approved extensions for teamspaces.
Support for Code Genie	The Code Genie feature now automates pull request workflows in Control repositories providing actionable insights, and automatically approving or rejecting PRs.
Build	
Support for the Build MCP server	You can now connect to the Build MCP server from external MCP clients such as VS Code and Claude Desktop to retrieve and update build resources directly from the external clients.
Test	
Support for downloading the agent from the Agents and Intercepts page	You can now download the DevOps Test Runtime from the **Agents and Intercepts** page to install and register the system as a local agent for playing back Web UI tests. See Installing and registering DevOps Test Runtime agents on page 1087.
Support for device emulation	You can now use the device mode of browsers to record and run tests or suites. Test Hub supports a set of device profiles for phones and tablets of four different sizes. Depending on the device size that you select, the device opens in Edge or Chrome browser. See Web UI test creation on page 769, Recording a Web UI test on page 771, Running server test assets on page 1321, Running server suite assets on page 1329, and Running server test assets on an agent on page 1324.
Support for running UI tests across multiple browsers in a single run	You can now run Web UI tests across multiple browsers in a single test run. This feature allows you to configure multiple browser environments and run the test against them simultaneously, helping you validate application behavior across different

Feature title	Description
	browsers more efficiently. See Running suites on multiple browsers and devices on page 839.
Support to define custom environments for test runs	You can now define custom environments to manage sets of variables such as application URLs and user credentials. When you run a test by using the environment, the variables defined in the environment dynamically replace the values within your test script. See Test environments on page 869, Creating an environment on page 870, and Modifying an environment on page 872.
Support for wildcard patterns to include or exclude multiple tests or suites for a test run	You can now use wildcard patterns to include or exclude multiple tests or suites by their name for a test run. You can configure the suite to select tests or suites by adding wildcard patterns. You can use a single asterisk (*) to match items within the same folder, or double asterisks (**) to match items in multiple subdirectories. See Running multiple suites or tests by using the wildcard pattern on page 837.
Support for individual results for test run as part of a suite	You can now view results for child tests that run as part of a parent suite. Each child test generates its own result, making it easier to review failures. These results are listed in the Results tab, where you can identify failed tests and analyze their results. See Test suite management on page 829 and Test results and reports on page 1501.
Support for viewing the run status of the asset during publishing	You can now view the run status of assets when reviewing changes on the **Changes** page before publishing. Each executable asset displays a colored status indicator in the parent row, so that you can view the most recent run result for that specific revision of the asset. See Publishing all changes in an Edit branch on page 1023.
Enhanced test repair experience	You can now repair tests by using the **Show Result** option. You can view the result details, including run status, duration, statistics, and reasons for failures. You can analyze results and verify step details without switching back to edit mode. See Viewing step-level results and updating tests on page 789.
Enhanced smartshots access to update tests	You can now access additional smartshots to update the test by using the **Replace Smartshot** option in the step details pane. The **Result Smartshots** section displays smartshots from the last test run, and the **Application Smartshots** section displays all previously captured smartshots for the application. See Viewing step-level results and updating tests on page 789.
Support for multiple dataset configurations in server assets	You can now configure multiple datasets for server test assets, such as Tests and Suites to assign dataset values to test variables during the test run. This feature supports dynamic data usage during test runs by allowing additional datasets to populate test variables, enabling subsequent steps to use unique and data-driven values. See Multiple datasets in tests and test suites on page 866.

What's New in DevOps Loop 2025.12 (2.0.0)

Release date: 12 January 2026

The following table lists the new features, enhancements, or other changes made in DevOps Loop:

Feature title	Description
GKE platform support	You can now install DevOps Loop on Google Kubernetes Engine, with full support for Filestore NFS, DNS configuration, load balancers, and Helm-based installation. See Installing DevOps Loop on Google Kubernetes Engine (GKE) on page 58.
Learning Loop enhancements	Learning Loop now provides an automated CI/CD demonstration flow from code changes to build, deployment, and a live updated webpage served through a new NgineX web server pod with a unique URL for each loop. See Learning Loop overview on page 151.
Support for DevOps Loop MCP server integration	You can now integrate the DevOps Loop MCP server with external MCP clients such as VS Code and Claude Desktop to retrieve and update loop resources directly from those tools. See Connecting Claude to the DevOps Loop MCP server on page 130.
Enhanced Loop Genie with multi-agent workflow engine	Loop Genie now uses a multi-agent workflow engine that can perform multi-step automation, run complex tasks, and produce well-formatted responses by using models such as **Claude Desktop**, **OpenAI**, and **Gemini**.
Support for a centralized Loop Details page	You can now view all loop configuration details in one place through the new **Loop Details** page, including linked resources and team members, with options to add additional test projects and repositories. See Loop Genie overview - Tech Preview on page 3277.
AI provider support for Loop Genie	You can now configure AI providers such as Claude Desktop and Gemini in DevOps Loop to enhance the AI capabilities of Loop Genie. See Configuring Claude integration on page 99, Configuring Gemini integration on page 101.
UI and accessibility enhancements	DevOps Loop UI is now improved with consistent color schemes, fonts, pagination, and alignment which provides a uniform and accessible user experience.
Enhanced search capabilities	The Search feature has been enhanced with new capabilities that make it easier to find and understand information across your loops. You can now search multiple artifact types including issues, builds, tool requests, commits, and deployments. You can also filter results by selecting a loop. See AI-powered search on page 156.
Support for mixed licensing tiers	As an administrator, you can now assign users to different licensing tiers: Essential, Standard, or Premium with access and feature availability automatically adjusted based on the selected tier. Non-admin users receive a tier during onboarding, and admin users automatically receive the highest tier available. See DevOps Loop licensing information on page 78.
Plan	

Feature title	Description
UI enhancements for query links and work item subscriptions	You can now copy query links to the clipboard and more easily subscribe other users to work items.
Enhancements to Schema Designer	You can now apply new schema revisions directly to your application, including adding and editing record-type Fields, Actions, and States.
Code	
Viewing metrics in the running dev containers	You can now get real-time visibility into key dev container resource metrics to help optimize performance. See Working with dev containers on page 169.
Support for automated SSH authentication by using the Open with DevOps Code UI option	You can now securely clone a Control Git repository by using the DevOps Code UI with SSH authentication. See Authenticating with DevOps Code UI on page 192.
Support for temporary GitHub Copilot in dev containers	You can now configure a dev container to support GitHub Copilot for AI-powered code suggestions and chat assistance. See Installing GitHub Copilot in dev containers on page 223.
Build	
Automatic Build agent configuration	Build agents are now automatically configured and added to the agent pool during the teamspace initialization.

What's New in DevOps Loop 2025.09 (1.0.3)

Release date: 31 October 2025

The following table lists the new features, enhancements, or other changes made in DevOps Loop.

Feature title	Description
Support for installation of DevOps Loop in an air-gapped environment	You can install DevOps Loop in environments without internet access. See Installation of DevOps Loop in an air-gapped environment on page 69.
Support for private CA and self-signed certificates	You can now configure DevOps Loop to trust private or self-signed certificates. See Private CA and self-signed certificate support on page 87.
Support for backup and restoration of DevOps Loop data	You can now back up and restore DevOps Loop data by using Velero and Minio for disaster recovery and upgrades. See Backup and restoration of DevOps Loop on page 84.
Support for upgrading DevOps Loop	You can now upgrade DevOps Loop with support for restoring configurations and resources if needed. See Upgrading DevOps Loop on page 90.

Feature title	Description
Support for Learning Loop with built-in sample data	You can now use sample data for Build, Control, Deploy, and Plan by enabling Learning Loop for training and experimentation without affecting production data. See Learning Loop overview on page 151.
Support for IBM watsonx integration for Loop Genie	You can now integrate IBM watsonx in DevOps Loop to enhance the AI capabilities of Loop Genie. See Configuring IBM watsonx integration on page 102.
Enhanced Loop Genie AI assistant - Tech Preview	You can now use Loop Genie as an intelligent, chat-based assistant to manage teamspace users, loops, and perform control actions such as creating branches or managing pull requests. New features also include voice interaction and improved tracking capabilities. See Loop Genie overview - Tech Preview on page 3277, Prompt references for Loop Genie on page 3286.
Support for customizing Dashboards	You can now edit dashboards, swap panels, clone panels, and add titles directly within DevOps Loop, which provides enhanced customization and visibility. See Dashboards and insights on page 154.
Support for Search across issues	You can now search for issues such as epics, defects, and stories across loops and teamspaces. Search results include issue metadata, associated URLs, and AI-generated summaries with results filtered by your access permissions. See AI-powered search on page 156.
Support for loop enabling or disabling functionality	You can now enable or disable loops. Disabling a loop moves it to a **Disabled** tab and stops associated plugins and webhooks, while re-enabling the loop restores all previously disabled functionality. See Disabling a loop on page 147 and Enabling a loop on page 147.
Support for Build integration	Build is now automatically set up during loop and teamspace creation, which streamlines resource setup and CI/CD steps.
Support for UI and accessibility enhancements	You can now experience improved UI consistency across DevOps Loop, including updated color schemes, fonts, pagination, and alignment which provides a uniform and accessible user experience.
Support for removing members from Teamspaces and Loops	You can now remove members from loops, which automatically removes them from all associated capabilties of DevOps Loop. After you remove members from loops, you can also remove the members from their teamspaces. See Removing a member from a loop on page 146 and Removing a member from a teamspace on page 135.
Measure	
Support for integration with Measure to export test execution metrics	You can now integrate Test Hub with Measure by using a webhook and specify additional parameters related to the test run so that you can view the summary metrics of the test run in the Measure dashboard. The results are categorized based on the type of test – Functional, Unit tests, API, and Performance.

Feature title	Description
Code	
Support for launching Dev Containers via URL	You can now launch Dev Containers directly from a shared URL in DevOps Code, not just from the landing page, enabling quicker access and simplified collaboration.
Support for viewing running Dev Containers	A new Running Containers tab is available on the landing page to show active Dev Containers in DevOps Code for improved visibility and management.
Display of Dev Container descriptions in UI	Each Dev Container tile on the landing page now displays its description, helping users understand the container's purpose before launching it.
Inclusion of sample Dev Containers in new teamspaces	Each newly created teamspace is now prepopulated with a set of sample Dev Containers, enabling users to explore and start development quickly without manual setup.
Support for Git device code authentication	You can now authenticate Git operations using device code authentication on page 196, ideal for secure or restricted environments.
Support for embedding preinstalled extensions in Dev Container images	As an admin, you can now embed extensions directly into Dev Container images, ensuring they are preinstalled at launch - ideal for air-gapped environments without access to online extension registries on page 181.

What's new in DevOps Loop 2025.06 (1.0.2)

The following section lists the new features, enhancements, or other changes made in DevOps Loop:

Feature title	Description
Support for teamspace creation	You can now create teamspaces for your teams in DevOps Loop and the linked teamspaces in all the integrated applications such as DevOps Plan, DevOps Control, DevOps Code, DevOps Test, DevOps Deploy, DevOps Measure, and DevOps Release. See Teamspace management on page 132, Creating a teamspace on page 132, and Adding or inviting members to a teamspace on page 134.
Support for OpenSearch dashboards	You can now view the visualization of the data that is gathered through the DevOps activities in the integrated applications. See Dashboards and insights on page 154.
Integration of AI providers - Tech Preview	You can now integrate AI providers such as OpenAI and Ollama to enable Loop Genie to search and summarize the project data in a loop. See AI provider integration for Loop Genie - Tech Preview on page 94, Configuring OpenAI integration on page 97, Configuring Ollama integration on page 103.
Integration of Loop Genie - Tech Preview	You can now use the Loop Genie chatbot to answer your queries related to the project in a loop. See Interacting with Loop Genie on page 3281.
Revoking licenses	You can now revoke licenses for users when they are no longer required in a project. See Revoking a license on page 81.

Feature title	Description
Integration of Build	You can now configure and use DevOps Build as part of the platform for managing continuous integration and build processes.
Support for custom dev containers	DevOps Code now supports custom dev containers. A teamspace owner can configure the dev containers accessible to the members. The configuration is based on the Dev Containers specification. You can push specifications of Dev containers to a Control Git repository to version the containers.
Support for a new landing page	DevOps Code now includes a new landing page. From the landing page, you can select the teamspace that you want to work and choose the branch from where you can fetch the available dev containers. You can launch, open, or terminate Dev containers directly from this landing page.

What's new in DevOps Loop 2025.03 (1.0.1)

The following section lists the features, enhancements, or other changes made in DevOps Loop:

Feature title	Description
Integration of DevOps Code	You can now use DevOps Code (Code) as part of the platform. Code is a cloud-based integrated development environment for creating and managing your code. See Code on page 159.

Defect fixes

You can find the defects that are fixed in this version of HCL DevOps Loop.

Defects fixed in DevOps Loop 2026.08 (2.0.2.1)

The defects fixed in DevOps Loop 2026.08 (2.0.2.1) are as follows:

ID	Description
NEXUS00006400	Previously, DevOps Code deployments in the AirGapLab environment failed with an image pull error because the Alpine image path was incorrect. This problem is fixed.
NEXUS00006417	Previously, repeated Google API references during Loop page loads caused UI slowness. This problem is fixed.
NEXUS00006553	Previously, external font API dependencies affected support for air-gapped environments. This problem is fixed.

Known issues

You can find the known issues that are identified in this version of HCL DevOps Loop.

Known issues in DevOps Loop 2026.08 (2.0.2.1)

The following known issue is identified in this release:

ID	Description
NEXUS00006751	When you create a teamspace by using an email ID that is exactly 12 characters long, the operation might fail at the Measure phase because the MongoDB 5.9.2 package incorrectly treats the email value as an ObjectId. In this case, the teamspace might be created in Velocity, but the overall teamspace creation task does not complete successfully. To work around this problem, you must avoid using email IDs that are exactly 12 characters long.
NEXUS00006817	When you run tests through Deploy in an HCL DevOps Loop in an air-gapped environment, the tests might fail because the HCL DevOps Test Hub plugin version 4.3 does not work correctly with the Test Hub integration. To work around this problem, navigate to Deploy and update the HCL DevOps Test Hub plugin to version 5.0 to run Test Hub tests successfully.

Known issues from earlier versions

The known issues identified in the earlier versions of DevOps Loop that are still applicable are as follows:

ID	Description	Identified in version	Resolved in version
NEXUS00006323	When you install IBM DevOps Loop version 2.0.2 by using the default IBM entitled registry, DevOps Code dev containers that use the GitHub Copilot extension image or Trivy images might fail to start because the image registry is not configured correctly for the generated image specification. To work around this problem, you must update the `IMAGE_REGISTRY` environment variable in the `devops-code-container-mgr` deployment after the installation is complete by following these steps: 1. From a workstation where the Kubernetes cluster is configured and the `kubectl` command is available, run the following command: `kubectl edit deployment devops-code-container-mgr -n devops-loop`	2.0.2	2.0.2.1

ID	Description	Identified in version	Resolved in version

	2. In the image deployment configuration, update the `IMAGE_REGISTRY` environment variable value with the following:
	`env:`
	`- name: IMAGE_REGISTRY`
	`  value: cp.icr.io/cp`
	3. Save the changes and exit the editor that opened when you ran the `kubectl edit` command.
	Kubernetes restarts the `devops-code-container-mgr` pod after the deployment configuration is updated.
	
	After the pod restrarts, you can create DevOps Code dev containers by using the GitHub Copilot extension image and Trivy images successfully. If you run the DevOps Loop installation script again, you must repeat this configuration change.
	
	Alternatively, before rerunning the installation, you must update the `ibm-devops-loop/charts/ibm-devopscode/templates/helper-deployment.yaml` template to set the `IMAGE_REGISTRY` value correctly.		

NEXUS0000 6321	The demo installation scripts that are included with DevOps Loop 2.0.2 contain incorrect configuration values. As a result, the scripts might fail or produce unexpected results unless they are corrected before execution.
	
	To work around this problem, you must make the following changes before running the demo installation scripts:
	1. Update the Loop chart version to the correct value:
	`LOOP_CHART_VERSION=2.0.200`
	2. Correct the `JSONPath` command by removing the extra closing brace (`)`).
	Replace:
	`-o jsonpath="{.data.valkey-password}" | base64 -d)`
	with
	`-o jsonpath="{.data.valkey-password}" | base64 -d`
	After making these changes, you must rerun the demo installation scripts.
		2.0.2	2.0.2.1

ID	Description	Identified in version	Resolved in version
NEXUS0005995	When you upgrade the DevOps Loop platform to version 2.0.2, existing 2.0.1 Learning Loops continue to use the Ecological Payments sample application and do not include the new Loop Genie tasks that are introduced in version 2.0.2. The **Upgrade Loop** option is not available for loops that were created in version 2.0.1. To work around this problem, you must create a new Learning Loop after upgrading to DevOps Loop 2.0.2 to use the **CycleShop** sample application and the latest Loop Genie tasks and tools.	2.0.2	
NEXUS0006310	When teamspace creation fails, and you select **Cancel and Retry Later**, the failed teamspace is expected to appear in a red ribbon for later retry. Instead, the page becomes unresponsive. To work around this problem, you must sign in with a different user account and create the Teamspace again.	2.0.2	2.0.2.1
NEXUS0006264	When you use Loop Genie to list the available loops, the request might fail because of a tool or domain metadata validation issue. As a result, Loop Genie might not display the available loops. Currently, there is no workaround for this problem.	2.0.2	
NEXUS0006216	When you use Loop Genie, the first request in a new session might fail with an internal server error because of API rate limiting. Although model validation completes successfully, the initial request might not be processed. To work around this problem, you must retry the request after a short interval. If the problem persists, you must start a new Loop Genie session and try the request again.	2.0.2	
NEXUS0006225	When you use Loop Genie in a long chat session, the conversation context might not be summarized correctly as the model context window approaches its limit. As a result, Loop Genie might not respond as expected or stop processing requests. To work around this problem, you must start a new Loop Genie session and continue your conversation there.	2.0.2	
NEXUS0	When a user is assigned an **unsupported Keycloak group and Measure/Release role combination**, the system does not display a	2.0.2	

ID	Description	Identified in version	Resolved in version
0005427	clear authorization error. Instead, the user might experience one of the following behaviors: • Repeated login redirect/loop during authentication • Inability to load Measure or Release pages • No explicit error message indicating invalid role mapping To work around this problem: 1. Ensure the user is assigned a supported Keycloak group. 2. Verify the user is mapped to a valid Measure/Release role combination. 3. For users added to the Users group in Keycloak, use the authorization feature in Measure or Release to assign the required permissions. 4. Re-login after correcting the configuration.		
NEXUS0005067	When you run the list test command, the response returns only the count of projects instead of detailed test information such as test names, IDs, or metadata. In some cases, the returned data might also be inaccurate. Also, when you run a test prompt, the system might return a **test asset file not found** error even when the correct project ID and test name are provided, indicating an issue in locating or mapping the specified test file. Currently, there is no workaround for this problem.	2.0.2	
NEXUS0006206	When you access the `/teamspaces` or `/teamspaces/{id}/tasks` API endpoints in an IHD environment, you might experience slow response times. Currently, there is no workaround for this problem.	2.0.2	
NEXUS0006316	The Jenkins data is not available in the **Deployment by Environment Name** chart in Analytics Dashboards. Currently, there is no workaround for this problem.	2.0.2	2.0.2.1

ID	Description	Identified in version	Resolved in version
NEXUS0000 4133	When you use IBM watsonX, it might not provide proper responses for multistep workflows. The issue is currently observed only with workflows involving multiple steps. Single-step workflows work as expected. To work around this problem, you must use IBM watsonX only for single-step workflows until the issue is resolved.	2.0.0	
NEXUS0000 3275	When an administrator is added to a loop by another administrator before logging into Measure for the first time or before creating their own Loop, the administrator cannot log in to Measure. To work around this problem, you must log in to Measure as another administrator (for example, the one who created the loop) and assign the global *Product Administrator* role to the administrator who cannot log in.	1.0.3	

Chapter 2. System Requirements for DevOps Loop 2026.08 (2.0.2.1)

This document includes information about hardware and software requirements for HCL DevOps Loop.

Contents

- Hardware on page 26
- Operating systems and containers on page 26
    - Bit version support on page 27
    - Operating systems on page 27
    - Container Platforms on page 27
- Host prerequisites on page 27
    - Installation on page 28
    - Licensing on page 28
    - Runtime environment on page 28
    - Web browsers on page 28
- Supported software on page 29
- Disclaimers on page 25

Disclaimers

This report is subject to the Terms of Use and the following disclaimers:

The information contained in this report is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability, non-infringement, and fitness for a particular purpose. In addition, this information is based on HCL's current product plans and strategy, which are subject to change by HCL without notice. HCL shall not be responsible for any direct, indirect, incidental, consequential, special or other damages arising out of the use of, or otherwise related to, this report or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from HCL or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of HCL software.

References in this report to HCL products, programs, or services do not imply that they will be available in all countries in which HCL operates. Product release dates and/or capabilities referenced in this presentation may change at any time at HCL's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Discrepancies found between reports and other HCL documentation sources may or may not be attributed to different publish and refresh cycles for this tool and other sources. Nothing contained in this report is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results. You assume sole responsibility for any results you obtain or decisions you make as a result of this report.

Notwithstanding the Terms of Use users of this site are permitted to copy and save the reports generated from this tool for such users own internal business purpose. No other use shall be permitted.

Hardware

You can find information about the hardware requirements for HCL DevOps Loop.

The following table lists the requirements to run DevOps Loop on the Kubernetes Service (K8S) or IBM Cloud Kubernetes Service (IKS) platform.

Resource	**Requirement**	**Notes**
Number of worker nodes	4	If you are using Ollama, worker nodes must have at least 32Gib memory.
Processor	8 CPUs per node	
Memory	64 GiB total In addition, you might require 200 MB for each DevOps Code Dev Container used by a user.	
Disk space	256 GiB In addition, you might require 20 GB for each DevOps Code Dev Container used by a user.	
Network	1 Gbps	

Related information

System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25

Operating systems and containers

You can find details about the supported operating systems and containers for HCL DevOps Loop.

Contents

- Bit version support on page 27
- Operating systems on page 27
- Container Platforms on page 27

Bit version support

Bitness	Description
64-Exploit	The product or part of the product runs as a 64-bit application in the 64-bit platforms listed as supported.

Operating systems

Operating system	Hardware	Bitness	Notes
RHEL 8.8	x86-64	64-Exploit	
RHEL 8.10	x86-64	64-Exploit	
RHEL 9.1	x86-64	64-Exploit	
RHEL 9.2	x86-64	64-Exploit	
RHEL 9.4	x86-64	64-Exploit	

Container Platforms

You can find details about the supported container platforms.

Platform	Version	Note
Kubernetes Service (K8S)	1.33	The storage class must support ReadWriteOnce (RWO) and ReadWriteMany (RWX).

Related information

System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25

Host prerequisites

You can find the prerequisites that support the operating capabilities for HCL DevOps Loop.

Contents

- Installation on page 28
- Licensing on page 28
- Runtime environment on page 28
- Web browsers on page 28

Installation

Supported software	Version
Helm CLI	3.17

Licensing

License Server	Version	Notes
My HCLSoftware (MHS)	Latest Cloud version	
HCL Local License Server	5.1	

Runtime environment

Supported software	Version	Notes
Helm CLI	3.17	Required locally for Helm.

Web browsers

Browser	Version
Google Chrome	Latest release
Microsoft Edge	Latest release
Mozilla Firefox	Latest release

Related information

System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25

Supported software

You can find information about the supported software in HCL DevOps Loop.

Supported software in DevOps Loop 2026.08 (2.0.2.1)

Supported software	Version
HCL DevOps Plan	3.0.801
Test	11.0.9
HCL DevOps Deploy	8.2.2.1
HCL DevOps Velocity	5.2.6
HCL DevOps Build	7.1.0.5

Supported software in DevOps Loop 2026.06 (2.0.2)

Supported software	Version
HCL DevOps Plan	3.0.8
Test	11.0.9
HCL DevOps Deploy	8.2.2
HCL DevOps Velocity	5.2.5
HCL DevOps Build	7.1.0.4

Related information

System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25

Chapter 3. Getting Started

HCL DevOps Loop is a platform designed to support the full DevOps lifecycle. The Getting Started section introduces key concepts and walks through the initial steps to help new users begin working with the platform.

Before exploring the capabilities or performing tasks, ensure that the DevOps Loop is installed and properly configured. See Installation of DevOps Loop on page 45.

Overview of DevOps Loop

HCL DevOps Loop is a unified platform that brings every stage of the software development lifecycle into a continuous feedback loop. It connects teams, processes, and applications across planning, coding, testing, releasing, deploying, and measuring to deliver software faster, with greater visibility and governance.

DevOps Loop is built on cloud-native technologies that support CI/CD at scale and enhance automation with Loop Genie. This AI-powered assistant that provides search results across applications in the Loop and accelerates workflows and helps accelerate delivery.

With dashboards and analytics, DevOps Loop provides real-time insights into progress, quality, and delays across the toolchain. These insights help teams track value creation, enhance decision-making, and continuously improve delivery.

Enterprises can install DevOps Loop on IBM Red Hat OpenShift, IBM Cloud Kubernetes Service (IKS), Kubernetes Service (K8S), Google Kubernetes Engine (GKE), Amazon Web Services (AWS), and Air-Gapped environments to achieve flexibility and scalability across environments.

In addition to its platform-wide features, DevOps Loop includes a suite of integrated capabilities such as **Plan, Code, Control, Build, Test, Release, Deploy, Measure, and Harbor**(if enabled) which work together to support every stage of the software development lifecycle.

For security and compliance, DevOps Loop offers role-based access control so administrators can manage permissions and enforce governance across projects.

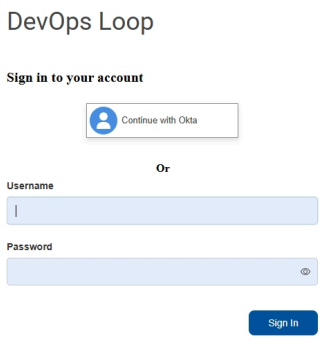
To learn more about the specific capabilities, see Capabilities of DevOps Loop on page 158.

User interface

You can find an overview of the HCL DevOps Loop user interface.

Login page

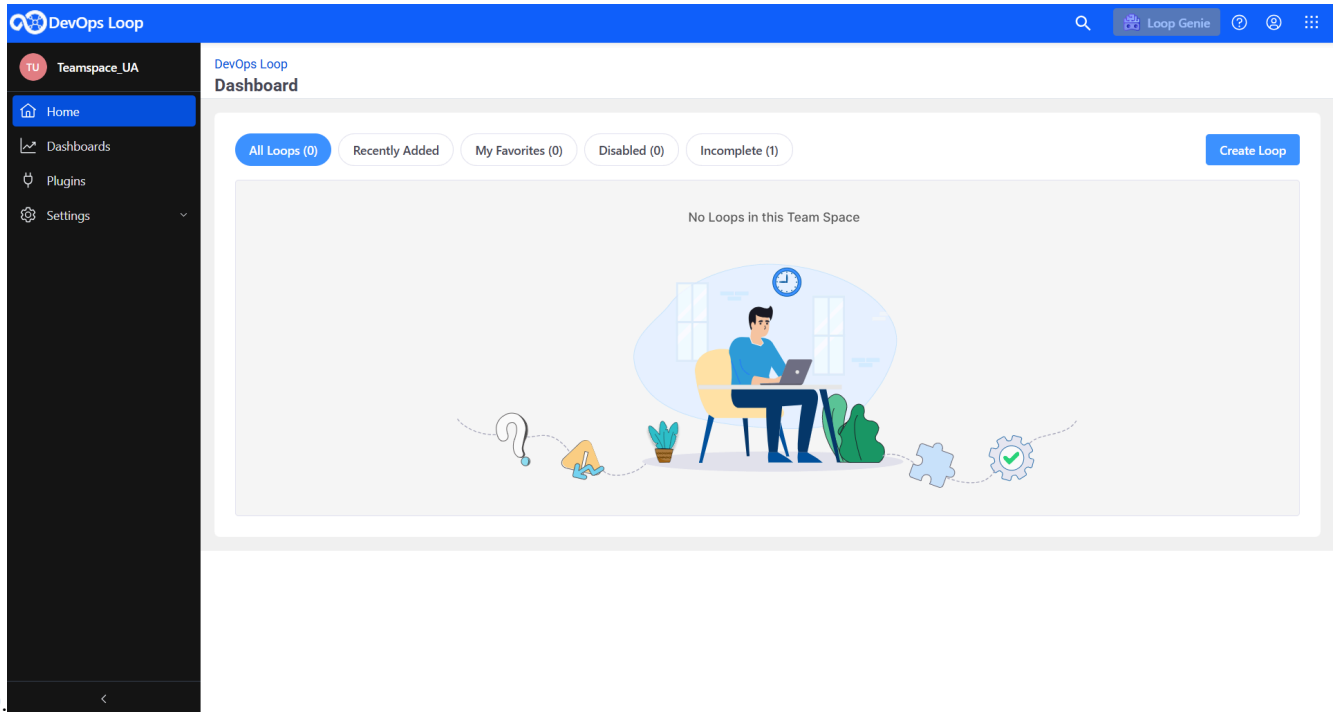
The login page provides an option to sign in to the DevOps Loop



platform.

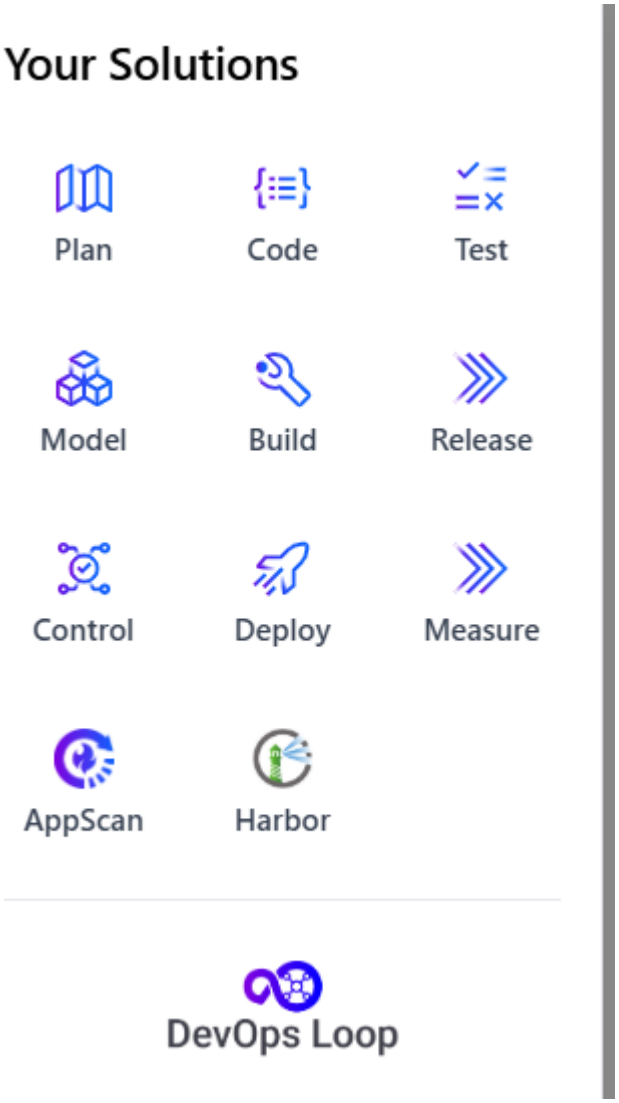
Home page

Within your Teamspace, you can create a loop to begin setting up your DevOps workflow. See Loop management on page 139.



Capability selector

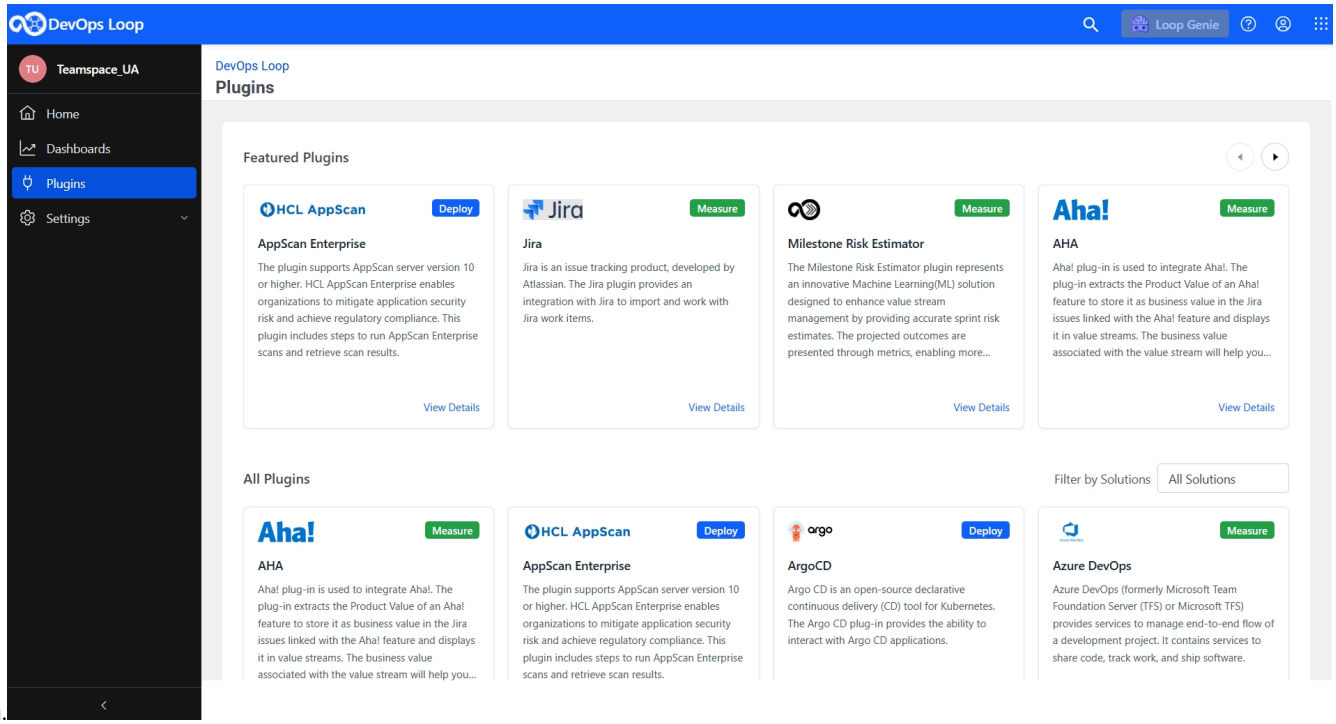
By using the capability selector icon , you can quickly access all DevOps Loop capabilities, such as Plan, Code, Test, Build, Release, Control, Deploy, Measure and Harbor (if enabled) from a single menu.



Plugins

The sidebar provides the **Plugins** option for you to navigate to the **Toolchain Engineering** page that lists the **Featured plugins** and **All plugins** sections. You can choose a plugin that is

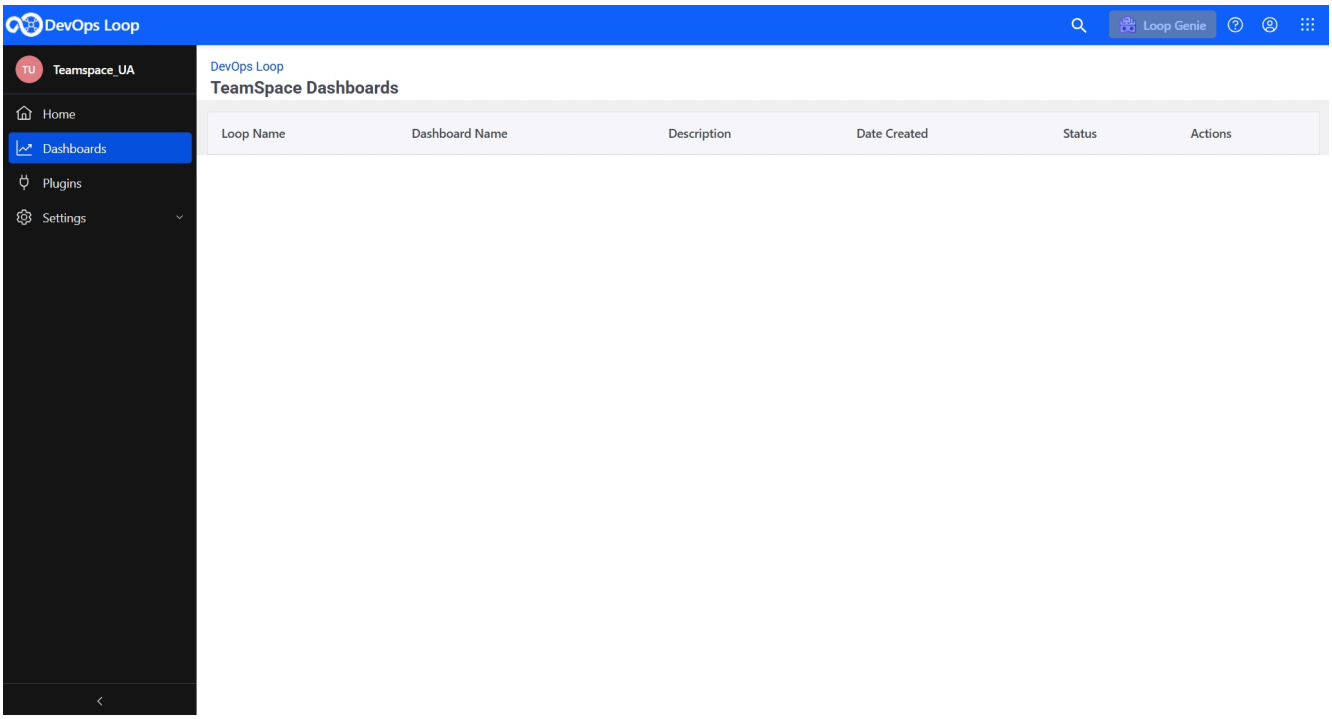
required for a specific solution from the **All plugins** section and click to download the



plugin.

Dashboards

The Dashboards page provides a centralized view where teams can access, filter, and manage all their DevOps Loop dashboards.



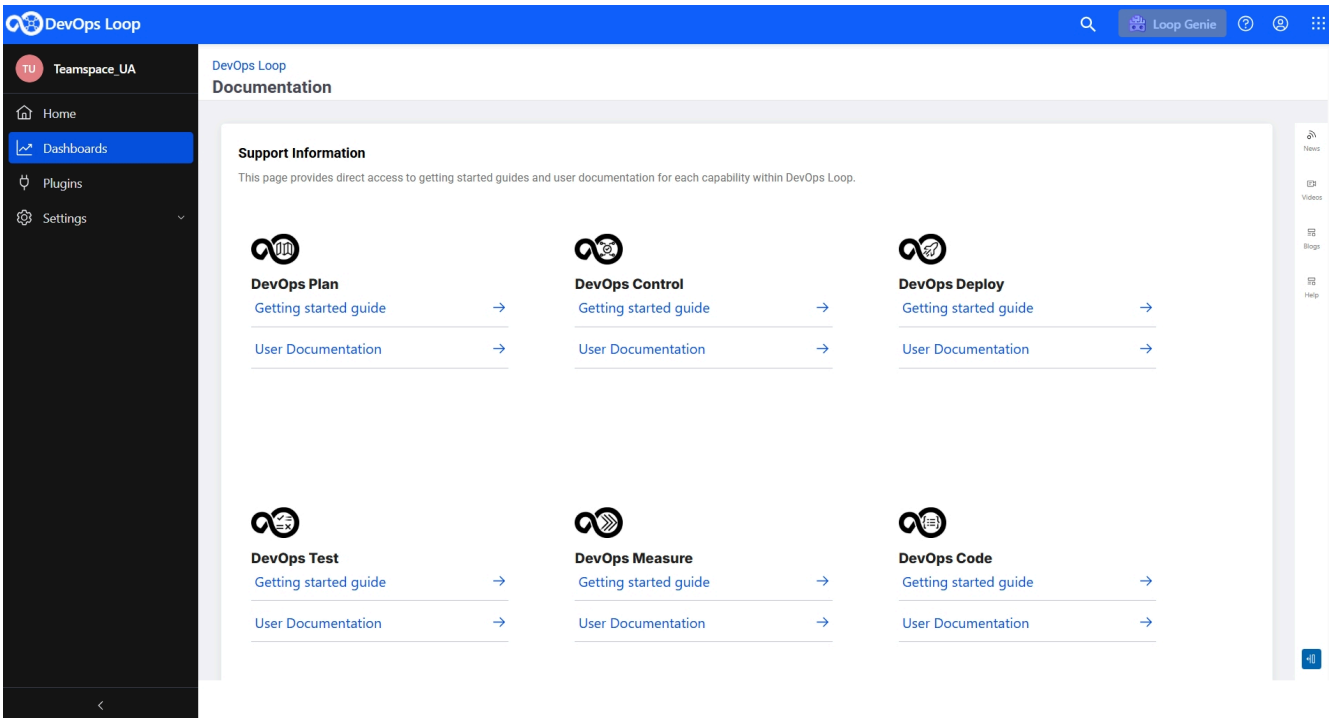
Settings

The sidebar provides you the option to administer users through the **User Administration** page and manage integrations through the **Installed Integrations** page. See Revoking a license on page 81 and Integrations in DevOps Loop on page 93.

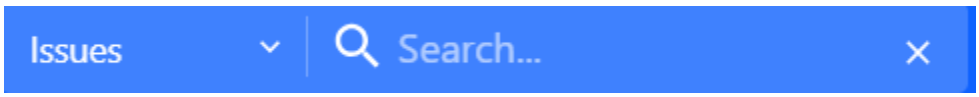
Help

You can click the **Help** icon to quickly access the product information and documentation resources.

You can select **Documentation** option in the **Help** menu to view getting-started guides and detailed user documentation for every capability within DevOps Loop.



Global Search

The search option 

helps you to find information across multiple capabilities. You can search across a wide set of artifact types, including issues, builds, tool requests, commits, and deployments, and you can filter results by loop or by category to quickly narrow your search.

Profile

The **Profile** displays the logged-in user's information and provides the option to log out of DevOps Loop.

To log out, you can click the profile icon , and then click **Logout**. The following logout page is displayed.

You can click **Login** to log in again.

Accessibility features

Accessibility features help users who have physical disabilities, such as visual and hearing impairment, or limited mobility, to use the software products successfully.

Accessibility compliance

The product documentation is published by using Oxygen XML WebHelp Responsive. To understand the accessibility compliance status for Oxygen XML WebHelp Responsive, refer to WebHelp Responsive VPAT Accessibility Conformance Report.

Accessing UI elements

DevOps Loop supports navigation in the UI by using different methods such as a mouse, keyboard, or touchpad.

You can use the keyboard keys such as **Tab**, arrow keys such as **UP**, **DOWN**, **LEFT**, and **RIGHT** to navigate to the different pages in the **Navigation** pane or to the different action labels in the right pane on the UI.

Chapter 4. Administration

This section provides information on administration, installation, configuration, and integrations and teamspace management.

Configuration of external databases for the capabilities in DevOps Loop

DevOps Loop capabilities use databases to store configuration data, execution results, and operational information. In the default configuration, some capabilities deploy internal databases as part of the Helm installation. However, you can configure the capabilities to use external databases that are managed outside the Kubernetes cluster.

By configuring external databases, you can use the existing enterprise database infrastructure, apply centralized backup and security policies, and manage the database lifecycle independently from the DevOps Loop platform.

Each capability in DevOps Loop supports external database configuration through Helm parameters or configuration files that are referenced during installation. You can configure the following capabilities to use external databases:

- **Plan** – Supports an external PostgreSQL database.
- **Control** – Supports an external PostgreSQL database.
- **Deploy** – Supports external relational databases such as MySQL, Db2, Oracle, or SQL Server.
- **Build** – Uses the same MySQL database configured for Deploy.
- **Test** – Supports an external PostgreSQL database.
- **Measure and Release** – Uses an external MongoDB database.

You can configure external databases for one or more capabilities by updating the DevOps Loop installation script and Helm configuration before running the installation.

If multiple capabilities require external databases, you can configure all required database settings in the installation script at the same time and run the DevOps Loop installation once. During installation, the Helm charts read the configured parameters and connect the corresponding capabilities to their external databases.

After the installation completes, the capabilities run in the Kubernetes cluster while using the specified external databases for storing application data.

Configuring an external database for Plan and Control

You can configure an external PostgreSQL database for the Plan capability in DevOps Loop and provide the database connection details in the Helm chart so that Plan connects to the external database instead of using the default PostgreSQL service included with the chart.

Before you begin

You must have completed the following tasks:

- Prepared the external PostgreSQL instance, and ensured that it is accessible from the Kubernetes cluster.
- Ensured that the database user has permissions for insert, update, delete, select, and schema modifications (create, alter, drop tables and indexes).
- Ensured that the Kubernetes cluster can reach the database host and port.
- Ensured that the following information is available:
  - `Database host name or IP address`
  - `Database port (default: 5432)`
  - `Database name`
  - `Database user name`
  - `Database password`

About this task

You must update the installation script so that Plan connects to the external database. The modifications include defining database variables, creating a Kubernetes secret for the database credentials, and passing the database parameters to Helm during installation.

1. Add the required external database variables at the beginning of the installation script:

   ```
   PLAN_DB_HOST=<database-host>
   PLAN_DB_PORT=5432
   PLAN_DB_NAME=plan_db
   PLAN_DB_USER=plan_user
   PLAN_DB_PASSWORD=<database-password>
   ```

   These variables define the connection details for the external PostgreSQL database used by Plan.

2. Set the Helm parameter `postgresql.enabled=false` to ensure the deployment does not create a database automatically.
3. Create a `devopsplan.yaml` file that defines the external database connection, tenant datastore, and disables the internal PostgreSQL service:

   ```
   ## Spring datastore settings (PostgreSQL)
   ibm-devopsplan-prod:
     spring:
       datastore:
         url: "jdbc:postgresql://${PLAN_DB_HOST}:${PLAN_DB_PORT}/${PLAN_DB_NAME}"
         username: ${PLAN_DB_USER}
         password: ${PLAN_DB_PASSWORD}

   ## Tenant datastore settings (PostgreSQL)
     tenant:
       datastore:
         server: ${PLAN_DB_HOST}
         dbname: ${PLAN_DB_NAME}
         username: ${PLAN_DB_USER}
         password: ${PLAN_DB_PASSWORD}

   ## Optional SMTP settings
     global:
   ```

```
    platform:
      smtp:
        sender: ${EMAIL_FROM_ADDRESS}
        host: ${EMAIL_SERVER_HOST}
        port: ${EMAIL_SERVER_PORT}
        username: ${EMAIL_SERVER_USERNAME}
        password: ${EMAIL_SERVER_PASSWORD}

  postgresql:
    enabled: false

## CONTROL external database settings
  control:
    postgresql:
      host: "CONTROL_DB_HOST"
      dbName: "CONTROL_DB_NAME"
      username: "CONTROL_DB_USER"
      password: "CONTROL_DB_PASSWORD"
    postgresql:
      enabled: false
```

4. Update the Helm install or upgrade command to use the external database by including the `devopsplan.yaml` file:

```
helm upgrade devops-loop oci://hclcr.io/launch-helm/hcl-launch-plan-prod \
  --install \
  --namespace ${NAMESPACE} \
  -f devopsplan.yaml
```

Results

After you run the installation script, DevOps Loop is installed, and the Plan capability connects to the external PostgreSQL database by using the parameters defined in the script and the `devopsplan.yaml` file

What to do next

You must install DevOps Loop to apply the configuration. See Installation of DevOps Loop on page 45.

Related information

Configuring external access to the Plan PostgreSQL database on page 44

Configuring an external database for Deploy

You can configure an external database for the Deploy capability in DevOps Loop and provide the database connection details in the Helm chart so that Deploy connects to the external database instead of creating one within the cluster.

Before you begin

You must have completed following tasks:

- Prepared the external PostgreSQL instance, and ensured that it is accessible from the Kubernetes cluster.
- Ensured that the database user has permission to access the database.
- Ensured that the Kubernetes cluster can reach the database host and port.
- Ensured that the required JDBC driver is available for the database.

- The following information is available:
  - `Database type (MySQL, Db2, Oracle, or SQL Server)`
  - `Database host name or IP address`
  - `Database port`
  - `Database name`
  - `Database user name`
  - `Database password`

About this task

You must update the installation script so that Deploy connects to the external database. The modifications include defining database variables, creating a Kubernetes secret for the database credentials, and passing the database parameters to Helm during installation.

1. Add the required external database variables to the beginning of the installation script:

   ```
   DB_TYPE=mysql
   DB_HOST=<database-host>
   DB_PORT=3306
   DB_NAME=deploydb
   DB_USER=dbuser
   DB_PASSWORD=<database-password>
   DB_SECRET=ucd-db-secret
   ```

   These variables define the connection details for the external database used by Deploy.

2. Create a Kubernetes secret for the database password.

   ```
   kubectl create secret generic ${DB_SECRET} \
     --namespace ${NAMESPACE} \
     --from-literal=dbpassword="${DB_PASSWORD}"
   ```

   The Deploy server reads the database password from this secret during initialization.

3. Set the Helm parameter `database.createDatabase=false` to ensure the deployment does not create a database automatically.
4. Update the `HELM_OPTIONS` in the script to include the external database connection parameters:

   ```
   HELM_OPTIONS="${HELM_OPTIONS} \
   --set hcl-launch-server-prod.database.type=${DB_TYPE} \
   --set hcl-launch-server-prod.database.hostname=${DB_HOST} \
   --set hcl-launch-server-prod.database.port=${DB_PORT} \
   --set hcl-launch-server-prod.database.name=${DB_NAME} \
   --set hcl-launch-server-prod.database.username=${DB_USER} \
   --set hcl-launch-server-prod.database.createDatabase=false \
   --set hcl-launch-server-prod.secret.name=${DB_SECRET} \
   --set hcl-launch-server-prod.database.fetchDriver=true"
   ```

Note: Add the following Helm parameter to ensure the Deploy server automatically downloads the required JDBC driver:

```
--set hcl-launch-server-prod.database.fetchDriver=true
```

Results

After you run the installation script, DevOps Loop is installed, and the Deploy capability connects to the external database by using the parameters defined in the script.

What to do next

You must install DevOps Loop to apply the configuration. See Installation of DevOps Loop on page 45.

Configuring an external database for Build

You can configure Build to use the same external MySQL database that is used by Deploy. When using an external database, the Build Helm chart must be configured to connect to the existing MySQL instance instead of creating a database inside the Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Ensured that the external MySQL database used by Deploy is available and accessible from the Kubernetes cluster.
- Ensured that the following information is available:
  - `MySQL host name or IP address`
  - `MYSQL port`
  - `Database name`
  - `Database user name`
  - `Database password`
- Ensured that the database user has permission to access the required database.
- Ensured that the Kubernetes cluster is connected to the MySQL server.
- Esnured that a Kubernetes secret containing the database password exists in the deployment namespace.

1. Add variables for the external MySQL database in the installation script:

   ```
   MYSQL_HOST=<mysql-host>
   MYSQL_PORT=3306
   DATABASE_NAME=<database-name>
   DATABASE_USER=<database-user>
   BUILD_DB_SECRET=build-secret
   ```

   These variables define the connection details for the external MySQL database used by the Build.
2. Create a Kubernetes secret that contains the database password by running the following command:

```
BUILD_DB_PASSWORD=<DATABASE_PASSWORD>
kubectl create secret generic build-secret -n devops-loop --from-literal=dbpassword="$BUILD_DB_PASSWORD"
```

#☺ **Important:** The `BUILD_DB_PASSWORD` must match the password assigned to the MySQL user.

3. Add the following Helm values to the `HELM_OPTIONS` section of the installation script. These parameters configure the Build Helm chart to use the external database.

```
--set hcl-devops-build.externalDB.enabled=true \
--set hcl-devops-build.externalDB.hostname=<MYSQL_HOST> \
--set hcl-devops-build.externalDB.port=<MYSQL_PORT> \
--set hcl-devops-build.externalDB.database=<DATABASE_NAME> \
--set hcl-devops-build.externalDB.user=<DATABASE_USER> \
--set hcl-devops-build.secret.name=build-secret
```

Results

After you run the installation script, DevOps Loop is installed and the Build capability connects to the external MySQL database by using the credentials stored in the Kubernetes secret.

What to do next

You must install DevOps Loop to apply the configuration. See Installation of DevOps Loop on page 45.

Configuring an external database for Test

You can configure an external PostgreSQL database for the Test capability in DevOps Loop and provide the database connection details in the Helm chart so that Test connects to the external database instead of using the PostgreSQL instance that is installed in the cluster by default.

Before you begin

You must have completed the following tasks:

- Prepared the external PostgreSQL instance, and ensured that it is accessible from the Kubernetes cluster.
- Ensured that the database user has permissions for insert, update, delete, select, and schema modifications (create, alter, drop tables and indexes).
- Ensured that the Kubernetes cluster can reach the database host and port.
- Ensured that the following information is available:
  - `Database host name or IP address`
  - `Database port (default: 5432)`
  - `Database name`
  - `Database user name`
  - `Database password`

About this task

You must update the installation script so that the Test capability connects to the external database. The modifications include defining database variables and passing the database parameters to Helm during installation.

1. Add the required external database variables to the beginning of the installation script:

```
TEST_DB_HOST=<database-host>
TEST_DB_PORT=5432
TEST_DB_NAME=testhubdb
TEST_DB_USER=postgres
TEST_DB_PASSWORD=<database-password>
TEST_CA_SECRET=<ca-secret-name>   # Optional, if using self-signed certificates
```

   These variables define the connection details for the external PostgreSQL database used by the Test.

2. Update the `HELM_OPTIONS` in the script to include the external database connection parameters:

```
HELM_OPTIONS="${HELM_OPTIONS} \
--set hcl-devops.postgresql.external=true \
--set hcl-devops.postgresql.hostname=${TEST_DB_HOST} \
--set hcl-devops.postgresql.port=${TEST_DB_PORT} \
--set hcl-devops.postgresql.username=${TEST_DB_USER} \
--set hcl-devops.postgresql.existingPassword=${TEST_DB_PASSWORD}"
```

   Note: By default the `sslmode` is set to `verify-full`.

   If the database uses a self-signed certificate, add the following parameter:

```
--set hcl-devops-prod.global.privateCaBundleSecretName=${TEST_CA_SECRET}
```

Results

After you run the installation script, DevOps Loop is installed and the Test capability connects to the external PostgreSQL database by using the parameters defined in the script.

What to do next

You must install DevOps Loop to apply the configuration. See Installation of DevOps Loop on page 45.

Configuring an external database for Measure and Release

You can configure Measure and Release to use an external MongoDB database by creating a Kubernetes secret to store the database connection URL and updating the Helm chart to reference this secret.

Before you begin

You must have completed following tasks:

- Ensured a running MongoDB instance is available.
- Ensured that the following information is available:
  - `MONGO_HOST`
  - `MONGO_PORT`
  - `Mongo_DATABASE`
  - `Mongo_USER`
  - `Mongo_PASSWORD`
- Ensured that the database user has the permission to access the required database.

1. Create a Kubernetes secret `mongodb-url-secret` in the target namespace, storing the MongoDB connection URL as the password.

```
MONGO_HOST=<mongo-host>
MONGO_PORT=3306
MONGO_DATABASE=<database-name>
MONGO_USER=<database-name>
MONGO_PASSWORD=<password>
MONGO_URL="mongodb://${MONGO_USER}:${MONGO_PASSWORD}@${MONGO_HOST}:${MONGO_PORT}/${MONGO_DATABASE}"

kubectl create secret generic mongodb-url-secret --namespace ${NAMESPACE} \
  --from-literal=password="${MONGO_URL}"
```

2. Add the following to the `HELM_OPTIONS` section of the installation script to reference the MongoDB secret:

```
--set hcl-ucv-prod.secrets.database=mongodb-url-secret
```

Results

After you run the installation script, DevOps Loop is installed, and the Measure and Release capabilities use the external MongoDB instance through the Kubernetes secret.

What to do next

You must install DevOps Loop to apply the configuration. See Installation of DevOps Loop on page 45.

Configuring external access to the Plan PostgreSQL database

You can configure external PostgreSQL access during installation by using a Helm upgrade.

About this task

When you install DevOps Loop by using the internal DevOps Plan PostgreSQL database and need to access it from DevOps Plan Designer or other supported PostgreSQL tools (such as pgAdmin 4), you must enable PostgreSQL access and configure the exposed port during installation or upgrade.

1. Use the following options with the Helm install or Helm Upgrade command:

```
--set hcl-devopsplan-prod.postgresql.enableExternalAccess=true \
--set hcl-devopsplan-prod.postgresql.service.exportPort=30106
```

2. Use the following details to connect to the PostgreSQL database:
    - **Host:** `global.domain`
    - **Port:** `30106`
    - **Database:** `postgres`
    - **Username:** `postgres`
3. Run the following command to retrieve the password:

```
kubectl get secret --namespace devops-loop devopsplan-postgresql \
-o jsonpath="{.data.tenant-datastore-password}" | base64 -d; echo
```

Note: You must ensure the namespace is set to `devops-loop`.

Results

You have configured external access to the Plan PostgreSQL database.

What to do next

You can view the connection details from the DevOps Plan UI. See Viewing connection details from the DevOps Plan UI on page 45.

Viewing connection details from the DevOps Plan UI

After you configured the external access to the Plan PostgreSQL, you can view the connection details from the DevOps Plan UI.

1. Open the **Tenant Management** page.
2. Select the required **Tenant**.
3. In the **TeamSpace** panel, select the **Details** tab.
4. Click **Show connection properties**.
5. View the PostgreSQL connection details for the selected TeamSpace.

Results

You have viewed the connection details from the DevOps Plan UI.

Installation of DevOps Loop

To get started working with HCL DevOps Loop, you must first install the platform in your environment.

You can install DevOps Loop on the following supported platforms:

- IBM Cloud Kubernetes Service (IKS)
- Kubernetes Service (K8S)
- Google Kubernetes Enigine (GKE)
- Amazon Web Services (AWS)
- Air-Gapped Environment

Note: Before you get started with the installation, review the requirements for configuring external databases for DevOps Loop capabilities. See Configuration of external databases for the capabilities in DevOps Loop on page 37.

Optional components

DevOps Loop supports optional components such as Harbor.

To enable Harbor during installation, set:

```
HARBOR_ENABLED=true
```

For prerequisites, configuration parameters, verification, and troubleshooting for Harbor, see Installing DevOps Loop with Harbor integration on page 107 and Troubleshooting Harbor integration.

For additional installation parameters, see Additional Helm parameters on page 76.

If you encounter issues during the installation or upgrade process, see Troubleshooting DevOps Loop installation and upgrade.

Installing DevOps Loop on IBM Cloud Kubernetes Service (IKS)

You can find information about the tasks that you can perform to install HCL DevOps Loop on IBM Cloud Kubernetes Service (IKS) by using a newly created cluster. You can use the Helm chart to perform the installation.

Before you begin

You must have completed the following tasks:

- Read and understood System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25.
- Installed the following CLI tools:
  - Kubectl
  - IBM Cloud CLI
- Installed Helm on the system from which you access the Kubernetes cluster. For more information, refer to Installing Helm.
- Set up a Kubernetes cluster. For more information, refer to Getting started with IBM Cloud Kubernetes Service.
- Set up the Secrets Manager in your Kubernetes Service cluster. For more information refer to IBM Cloud Secrets Manager.
- Obtained valid public certificates issued by trusted Certificate Authorities (CAs). Also, read and understood about managing certificates and secrets. For more information refer to Managing TLS and non-TLS certificates and secrets.

1. Navigate to **Cluster Management > Clusters > Overview > Actions > Connect via CLI** in your IBM Cloud account and associate the kubectl context with your cluster.
2. Navigate to **Cluster Management > Clusters > Ingress > Ingress Controllers (ALB)** and note down the IP address of the public ingress controller.

3. Navigate to **Cluster Management > Clusters > Ingress > Domain** and note down the region in the domain name of your cluster, which is in the format, `<cluster_name-id>.<region>.containers.appdomain.com`.
4. Click **Create** in the **Domain** tab and perform the following steps:
    a. Provide a domain name in the format, `<custom_name>.<region>.containers.appdomain.com`, by using the region noted down in the previous step.
    b. Provide the IP address of the public ingress controller noted down in step 2 on page 46.
    c. Set the domain as the default domain.

    Note: If you do not set this as the default domain, you must delete and re-create the domain to set it as the default domain.

5. Make the certificate and key available as a Kubernetes secret in the Kubernetes namespace that you use for DevOps Loop.

    Note: If a helm lookup error occurs on a secret, likely the certificate was not imported to the Loop namespace.

    To correct this error, import the certificate into your namespace: `ibmcloud ks ingress secret create --cluster <cluster_name_or_ID> --cert-crn <crn> --name <secret_name> --namespace <namspace>`

    You must use the IBM Secrets Manager to manage the life cycle of the certificate. You must also note down the name of the secret that contains the TLS certificate and key. For more information, refer to https://cloud.ibm.com/docs/containers?topic=containers-secrets&interface=ui#tls-custom.
6. Perform the following steps to install Emissary-ingress in your cluster:
    a. Run the following commands to set the Ambassador Edge Stack Helm chart:

    ```
    helm repo add datawire https://app.getambassador.io
    helm repo update
    ```

    b. Run the following commands to create a namespace and install the Ambassador Edge stack:

    ```
    kubectl create namespace emissary && \
    kubectl apply -f https://app.getambassador.io/yaml/emissary/3.9.1/emissary-crds.yaml
    kubectl wait --timeout=90s --for=condition=available deployment emissary-apiext -n
     emissary-system
    ```

    c. Run the following command to create `emissary-ports.yaml`:

    ```
    cat <<EOF > emissary-ports.yaml
    service:
      ports:
        - name: https
          port: 443
          targetPort: 8443
          #nodePort: <optional>
        - name: http
          port: 80
          targetPort: 8080
          #nodePort: <optional>
        - name: deploy-wss
    ```

```
      port: 7919
      targetPort: 7919
      #nodePort: <optional>
    - name: control-ssh
      port: 9022
      targetPort: 9022
      #nodePort: <optional>
EOF
```

d. Install Emissary-ingress by running the following command:

```
helm install emissary-ingress --namespace emissary datawire/emissary-ingress -f
 emissary-ports.yaml && \
kubectl -n emissary wait --for condition=available --timeout=90s deploy
 -lapp.kubernetes.io/instance=emissary-ingress
```

7. Note down the domain name for your cluster, which is in the format, `<custom_name>.<region>.containers.appdomain.com`.
   You can navigate to **Containers > Cluster Management > Clusters > Ingress > Domains** in the IBM Cloud console or run the following command to list the domains in your cluster:

```
ibmcloud ks ingress domain ls --cluster <CLUSTER_NAME>
```

8. Perform the following steps to access the HCL Harbor container registry:
   a. Get a key to the HCL Harbor container registry.
   b. Log in to HCL Harbor container registry with the HCL ID and password that are associated with the entitled software.
   c. Copy the pre-generated CLI secret from the **User Profile** page.
   d. Create the following three secrets in the target namespace to pull images from the HCL Harbor container registry:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

Note: Secrets are namespace-specific and they are required to install DevOps Plan.

9. Run the following command to view the `README.md` file:

```
helm show readme oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201
```

10. Optional:
11. Update the following parameters and the other required parameters in the script in the Helm README with the correct values:
    - DOMAIN
    - TLS_CERT_SECRET_NAME
    - RWO_STORAGE_CLASS=nfs-client
    - RWX_STORAGE_CLASS=nfs-client

    For DOMAIN and TLS_CERT_SECRET_NAME, you must provide the values noted down in the previous steps.

12. Run the script in the Helm README for K8 installation.

    Note: If the installation fails due to timeout, run the installation script again.

13. Perform the following steps to enable non-HTTP and additional special services:
    a. Run the following command to display the IP of the L4 load balancer installed as part of DevOps Loop:

    ```
    kubectl get svc --namespace emissary emissary-ingress -o
     jsonpath='{.status.loadBalancer.ingress[0].ip}'
    ```

    b. Perform the following steps to create a new domain by using the IP of the L4 load balancer:
        i. Navigate to **Cluster Management > Clusters > Ingress > Domains >** .
        ii. Copy the default domain name.
        iii. Click **Create**.
        iv. Enter `service-<copied_domain_name>` in **Name**.
        v. Enter the IP of the L4 load balancer in **IP address**.
        vi. Click **Create**.

    You can also run the following command from CLI:

    ```
    ibmcloud ks ingress domain create --cluster CLUSTER [--crn CRN] [--is-default] [--domain
     DOMAIN] [--hostname HOSTNAME] [--ip IP] [--output OUTPUT] [--domain-provider PROVIDER] [-q]
     [--secret-namespace NAMESPACE] [--zone ZONE]
    ```

    Note: You must ensure that the domain created in this step is not the default domain.

Results

You have installed DevOps Loop on IKS by using a newly created cluster.

Installing DevOps Loop on Kubernetes Service (K8S)

You can find information about the tasks that you can perform to install HCL DevOps Loop on Kubernetes Service (K8S). You can use the Helm chart to perform the installation.

Before you begin

You must have completed the following tasks:

- Read and understood System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25.
- Installed the Kubernetes CLI tool, Kubectl.
- Installed Helm on the system from which you access the Kubernetes cluster. For more information, refer to Installing Helm.
- Set up a Kubernetes cluster. For more information, refer to Kubernetes Documentation.
- Read and understood administering a cluster and managing TLS certificates in a cluster. For more information, refer to Administer a Cluster and Manage TLS Certificates in a Cluster.
- Set up the cert-manager in your Kubernetes cluster. For more information refer to Kubernetes documentation.

About this task

You can select one of the following methods to install DevOps Loop on K8S:

- Installing on a K8S cluster that has load balancer resources available on page 50
- Installing on a K8S cluster with an upstream L7 load balancer on page 52
- Installing on a K8S cluster that has an upstream L7 load balancer and expects data to be re-encrypted on page 54

Installing on a K8S cluster that has load balancer resources available

Before you begin

You must have completed the following tasks:

- Ensured your cluster supports L4 load balancer resources.
- Ensured that an external fully qualified domain name with a certificate signed by a well-known CA or an intermediary is available.

1. Associate the kubectl context with your cluster by using the following commands:

```
kubectl config set-context <context_name> --namespace=<namespace-name> --cluster=<cluster-name>
 --user=<user-name>
kubectl config use-context <context-name>
```

2. Obtain the certificate and key for the domain that you use for DevOps Loop.
3. Make the certificate and key available as a Kubernetes secret in the Kubernetes namespace that you use for DevOps Loop.

   You should use the cert-manager or any standard mechanism to manage the life cycle of the certificate. You must also note down the name of the secret that contains the TLS certificate and key.
4. Perform the following steps to install Emissary-ingress in your cluster:
   a. Run the following commands to set the Ambassador Edge Stack Helm chart:

```
helm repo add datawire https://app.getambassador.io
helm repo update
```

   b. Run the following commands to create a namespace and install the Ambassador Edge stack:

```
kubectl create namespace emissary && \
kubectl apply -f https://app.getambassador.io/yaml/emissary/3.9.1/emissary-crds.yaml
kubectl wait --timeout=90s --for=condition=available deployment emissary-apiext -n
 emissary-system
```

   c. Perform the following step to create emissary-ports.yaml:

```
cat <<EOF > emissary-ports.yaml
service:
  ports:
    - name: https
      port: 443
      targetPort: 8443
      #nodePort: <optional>
    - name: http
      port: 80
      targetPort: 8080
      #nodePort: <optional>
    - name: deploy-wss
      port: 7919
```

```
      targetPort: 7919
      #nodePort: <optional>
     - name: build-wss
      port: 7920
      targetPort: 7920
      #nodePort: <optional>
    - name: control-ssh
      port: 9022
      targetPort: 9022
      #nodePort: <optional>
EOF
```

d. Install Emissary-ingress:

```
helm install emissary-ingress --namespace emissary datawire/emissary-ingress -f
 emissary-ports.yaml && \
kubectl -n emissary wait --for condition=available --timeout=90s deploy
 -lapp.kubernetes.io/instance=emissary-ingress
```

5. Open the ports in your firewall to the external ports and the node ports configured in the previous step.
   You can run the following command to determine the node ports if they are configured automatically:

```
kubectl get svc emissary-ingress --namespace emissary -o jsonpath='{range .spec.ports[*]}{.name}:
 {.nodePort}{"\n"}{end}'
```

6. Perform the following steps to access the HCL Harbor container registry:
   a. Get a key to the HCL Harbor container registry.
   b. Log in to HCL Harbor container registry with the HCL ID and password that are associated with the entitled software.
   c. Copy the pre-generated CLI secret from the **User Profile** page.
   d. Create the following three secrets in the target namespace to pull images from the HCL Harbor container registry:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

Note: Secrets are namespace-specific and they are required to install DevOps Plan.

7. Run the following command to view the `README.md` file:

```
helm show readme oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201
```

8. Update the following parameters and the other required parameters in the script in the Helm README with the correct values:
   - DOMAIN
   - TLS_CERT_SECRET_NAME
   - RWO_STORAGE_CLASS=nfs-client
   - RWX_STORAGE_CLASS=nfs-client

   For DOMAIN and TLS_CERT_SECRET_NAME, you must provide the values noted down in the previous steps.

9. Run the script in the Helm README for K8 installation.

   Note: If the installation fails due to timeout, run the installation script again.

10. Perform the following steps to enable non-HTTP and additional special services:
    a. Configure the DNS to route traffic from a second FQDN that is service-<DOMAIN> to the L4 load balancer that you created as a prerequisite.

       The DOMAIN value is the same as the value used in the helm chart.

    b. Configure your L4 node balancer to forward the ports configured in 4.c on page 50 and determined in 5 on page 51 to your cluster.

Installing on a K8S cluster with an upstream L7 load balancer

Before you begin

You must have completed the following tasks:

- Ensured that the external L7 load balancer and cluster support for L4 load balancer resources are available.
- Ensured that an external fully qualified domain name with a certificate signed by a well-known CA or an intermediary is available.

1. Perform the following steps to install Emissary-ingress in your cluster:
   a. Run the following commands to set the Ambassador Edge Stack Helm chart:

   ```
   helm repo add datawire https://app.getambassador.io
   helm repo update
   ```

   b. Run the following commands to create a namespace and install the Ambassador Edge stack:

   ```
   kubectl create namespace emissary && \
   kubectl apply -f https://app.getambassador.io/yaml/emissary/3.9.1/emissary-crds.yaml
   kubectl wait --timeout=90s --for=condition=available deployment emissary-apiext -n
    emissary-system
   ```

   c. Perform the following step to create emissary-ports.yaml:

   ```
   cat <<EOF > emissary-ports.yaml
   service:
     type: LoadBalancer  #Set to NodePort when using an external L4 load balancer
     ports:
       - name: http
         port: 80
         targetPort: 8080
         #nodePort: <optional>
       - name: deploy-wss
         port: 7919
         targetPort: 7919
         #nodePort: <optional>
       - name: control-ssh
         port: 9022
         targetPort: 9022
   ```

```
      #nodePort: <optional>
EOF
```

d. If the support for load balancer resources is not available in your cluster, edit the `emissary-ports.yaml` to change the type to NodePort.

An external L4 load balancer is required in this installation scenario.

e. Install Emissary-ingress:

```
helm install emissary-ingress --namespace emissary datawire/emissary-ingress -f
 emissary-ports.yaml && \
kubectl -n emissary wait --for condition=available --timeout=90s deploy
 -lapp.kubernetes.io/instance=emissary-ingress
```

2. Open the ports in your firewall to the node ports configured in the previous step.
You can run the following to determine the node ports if they are configured automatically:

```
kubectl get svc emissary-ingress --namespace emissary -o jsonpath='{range .spec.ports[*]}{.name}:
 {.nodePort}{"\n"}{end}'
```

3. Perform the following steps to access the HCL Harbor container registry:
   a. Get a key to the HCL Harbor container registry.
   b. Log in to HCL Harbor container registry with the HCL ID and password that are associated with the entitled software.
   c. Copy the pre-generated CLI secret from the **User Profile** page.
   d. Create the following three secrets in the target namespace to pull images from the HCL Harbor container registry:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

Note: Secrets are namespace-specific and they are required to install DevOps Plan.

4. Run the following command to view the `README.md` file:

```
helm show readme oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201
```

5. Update the following parameters and the other required parameters in the script in the Helm README with the correct values:
   - DOMAIN
   - RWO_STORAGE_CLASS=nfs-client
   - RWX_STORAGE_CLASS=nfs-client

   For DOMAIN, you must provide the values noted down in the previous steps.

6. Add the following parameter to the ADDITIONAL_HELM_OPTIONS section:

```
--set platform.emissary.l7Depth=<number_of_hops_to_load_balancer>
```

By default the value is set to 0, which indicates that there is no upstream load balancer. You must set the value to 1 for a single hop to a direct upstream load balancer.

7. Run the script in the Helm README for K8 installation.
8. Perform the following steps to enable non-HTTP and additional special services:
    a. If the load balancer resources are available in your cluster, then run the following command to determine the IP of the L4 load balancer:

    ```
    kubectl get svc --namespace emissary emissary-ingress -o
     jsonpath='{.status.loadBalancer.ingress[0].ip}'
    ```

    If the load balancer resources are not available, then configure an external L4 load balancer to open the non-http/https ports in the `emissary-ingress.yaml` to direct traffic to your cluster.

    b. Configure the DNS to route traffic from a second FQDN that is service-<DOMAIN> to the L4 load balancer that you created as a prerequisite.

    A certificate is not required for this domain. The DOMAIN value is the same as the value used in the helm chart.

Installing on a K8S cluster that has an upstream L7 load balancer and expects data to be re-encrypted

Before you begin

You must have completed the following tasks:

- Ensured that the external L7 load balancer, and cluster support for L4 load balancer resources or an external L4 load balancer are available.
- Ensured that an external fully qualified domain name with a certificate signed by a well-known CA or a self-signed certificate is available as required.

1. Obtain a certificate and key that covers all the nodes in your cluster through a Subject Alternative Name (SAN).
2. Configure your L7 load balancer to trust the certificate for the nodes in your cluster.

    Note: A health check endpoint is available at `/automation/healthz` for your load balancer to reference for health checks.

3. Make the certificate and key available as a Kubernetes secret in the Kubernetes namespace that you use for DevOps Loop.
    You must use the cert-manager or any standard mechanism to manage the life cycle of the certificate. You must also note down the name of the secret that contains the TLS certificate and key.
4. Perform the following steps to install Emissary-ingress in your cluster:
    a. Run the following commands to set the Ambassador Edge Stack Helm chart:

    ```
    helm repo add datawire https://app.getambassador.io
    helm repo update
    ```

b. Run the following commands to create a namespace and install the Ambassador Edge stack:

```
kubectl create namespace emissary && \
kubectl apply -f https://app.getambassador.io/yaml/emissary/3.9.1/emissary-crds.yaml
kubectl wait --timeout=90s --for=condition=available deployment emissary-apiext -n
 emissary-system
```

c. Perform the following step to create emissary-ports.yaml:

```
cat <<EOF > emissary-ports.yaml
service:
  type: LoadBalancer # NodePort if no LoadBalancer resources are available in your cluster
  ports:
    - name: https
      port: 443
      targetPort: 8443
      #nodePort: <optional unused if type Nodeport>
    - name: http
      port: 80
      targetPort: 8080
      #nodePort: <optional unused if type Nodeport>
    - name: deploy-wss
      port: 7919
      targetPort: 7919
      #nodePort: <optional unused if type Nodeport>
    - name: control-ssh
      port: 9022
      targetPort: 9022
      #nodePort: <optional unused if type Nodeport>
EOF
```

d. If no load balancer resources are available in your cluster, edit the `emissary-ports.yaml` to change the type to NodePort.

An external L4 load balancer is required in this installation scenario.

e. Install Emissary-ingress:

```
helm install emissary-ingress --namespace emissary datawire/emissary-ingress -f
 emissary-ports.yaml && \
kubectl -n emissary wait --for condition=available --timeout=90s deploy
 -lapp.kubernetes.io/instance=emissary-ingress
```

5. Open the ports in your firewall to the node ports configured in the previous step.
   You can run the following to determine the node ports if they are configured automatically:

```
kubectl get svc emissary-ingress --namespace emissary -o jsonpath='{range .spec.ports[*]}{.name}:
 {.nodePort}{"\n"}{end}'
```

6. Perform the following steps to access the HCL Harbor container registry:
   a. Get a key to the HCL Harbor container registry.
   b. Log in to HCL Harbor container registry with the HCL ID and password that are associated with the entitled software.
   c. Copy the pre-generated CLI secret from the **User Profile** page.
   d. Create the following three secrets in the target namespace to pull images from the HCL Harbor container registry:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

Note: Secrets are namespace-specific and they are required to install DevOps Plan.

7. Run the following command to view the `README.md` file:

```
helm show readme oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201
```

8. Update the following parameters and the other required parameters in the script in the Helm README with the correct values:
    - DOMAIN
    - TLS_CERT_SECRET_NAME
    - RWO_STORAGE_CLASS=nfs-client
    - RWX_STORAGE_CLASS=nfs-client

   For DOMAIN and TLS_CERT_SECRET_NAME, you must provide the values noted down in the previous steps.

9. Run the script in the Helm README for K8 installation.
10. Perform the following steps to enable non-HTTP and additional special services:
    a. If the load balancer resources are available in your cluster, then run the following command to determine the IP of the L4 load balancer installed as part of DevOps Loop:

```
kubectl get svc --namespace emissary emissary-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

    If the load balancer resources are not available, then configure an external L4 load balancer to open the non-http/https ports in `emissary-ingress.yaml` to direct traffic to your cluster.

    b. Configure the DNS to route traffic from a second FQDN that is service-<DOMAIN> to the L4 load balancer that you created as a prerequisite.

    A certificate is not required for this domain. The DOMAIN value is the same as the value used in the helm chart.

Installation of DevOps Loop on Google Kubernetes Engine (GKE)

You can find information about tasks that you can perform to install DevOps Loop on Google Cloud Platform (GCP) by using the Google Kubernetes Engine (GKE) service.

Related information

Preparing GCP infrastructure for installing DevOps Loop on GKE on page 57

Installing DevOps Loop on Google Kubernetes Engine (GKE) on page 58

Preparing GCP infrastructure for installing DevOps Loop on GKE

You must set up the required Google Cloud Platform (GCP) infrastructure including DNS, a global static IP address, and RWX Storage, before deploying DevOps Loop on Google Kubernetes Engine (GKE).

Before you begin

You must have completed the following tasks:

- Ensured that your GCP environment includes the required roles and APIs:
  - The service account used for the GKE cluster must have the `roles/file.serviceAgent` IAM role.
  - Enabled the following APIs at the project level:
    - `file.googleapis.com`
    - `gkehub.googleapis.com`

      These APIs and permissions are required for filestore provisioning, CSI driver operations, and certificate management within GKE.

1. Perform the following steps to configure DNS and network resources:
   a. Create a host name in Cloud DNS for the DevOps Loop application.
   b. Reserve a **global static IPv4 address** in the same GCP project.
   c. Create a DNS A record that maps the domain name to the global static IP address.

   For example:
   - Name: `<host-name>`
   - Type: `A`
   - Value: `<GLOBAL_STATIC_IP>`

   Note: DNS must resolve and must be visible from the public internet or at least from GCP checkers for GKE-managed certificate provisioning to succeed.

2. Perform the following steps to configure `RWX StorageClass` for DevOps Loop:

   a. Create a custom `RWX StorageClass` with the following requirements:

      - Minimum 1 TiB storage
      - Provisioner: `filestore.csi.storage.gke.io`
      - VPC: must match the VPC network used by your GKE cluster

   Note: A custom `StorageClass` is required so filestore volumes are created within your cluster's VPC. Otherwise, GCP-provided filestore classes attempt provisioning under the default network.

3. Verify available `StorageClasses` by running the following command:

```
kubectl get storageclass
```

Results

You have prepared the required Google Cloud Platform (GCP) infrastructure for deploying DevOps Loop on Google Kubernetes Engine (GKE).

What to do next

You can now proceed with installing DevOps Loop on GKE.

Installing DevOps Loop on Google Kubernetes Engine (GKE)

You can install HCL DevOps Loop on a newly created GKE cluster by using the Helm chart.

Before you begin

You must have completed the following tasks:

- Read and understood System Requirements for DevOps Loop 2026.08 (2.0.2.1) on page 25.
- Gained access to the cluster and installed the following tools:
  - Kubectl
  - Google Cloud SDK and Google Cloud CLI
- Installed Helm on the system from which you access the Kubernetes cluster. For more information, refer to Installing Helm.
- Set up a Kubernetes (GKE) cluster under the platform - Google Kubernetes Engine (GKE) and Image type - Container-Optimized OS with containerd (cos_containerd). For more information, refer to Create a cluster and deploy a workload in the Google Cloud console.

  Note: Ensured that you have enabled the Filestore CSI driver and Cloud Storage Fuse CSI driver while creating the cluster to use the GCP CSI driver storage class.

- Prepared the required infrastructure in Google Cloud Platform (GCP). See Preparing GCP infrastructure for installing DevOps Loop on GKE on page 57.
- Ensured that you have the following details:
  - **DOMAIN (FQDN)** – the hostname you created in Cloud DNS for the Loop installation.
  - **STATIC_IP_ADDRESS** – the global static IP mapped to the DNS A record.
  - **NAMESPACE** – the target Kubernetes namespace (for example: `devops-loop`).

1. Run the following command to download the `GKE-Infra-Setup.sh` script from the Helm chart to provision and configure the load balancer with a Google managed certificate.

   ```
   (TMP_DIR=$(mktemp -d) && \
   helm pull hcl-helm/hcl-devops-loop --untar --untardir "$TMP_DIR" --version 2.0.201 > /dev/null 2>&1 && \
   sh "$TMP_DIR/hcl-devops-loop/scripts/GKE/GKE-Infra-Setup.sh" && \
   rm -rf "$TMP_DIR")
   ```

   This command downloads and copies, and runs the `GKE-Infra-Setup.sh` script in your temporary directory.

After you run the script, wait for certificate provisioning to complete. This process can take 10–60 minutes, depending on your cluster and DNS configuration.

2. Perform the following steps to access the IBM Entitled Registry:
    a. Log in to the **My IBM Container Software Library** using the ID and password associated with your entitled software.
    b. Navigate to the Entitlement keys section and select **Copy key** to copy your IBM Entitled Registry entitlement key to you clipboard.
    c. Create a secret in the target namespace named `hcl-entitlement-key` to pull images from the HCL Harbor container registry:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

3. Run the following command to add the repository:

```
helm repo add hclsoftware
https://us-artifactory1.nonprod.hclpnp.com/artifactory/hclsoftware-helm-dev --force-update
```

4. Run the following command to view the `README.md` file:

```
helm show readme hclsoftware-helm/hclsoftware-devops-loop
```

5. Update the following parameters in the Helm installation script with the correct values:
    - DOMAIN - the FQDN created in Cloud DNS
    - TLS_CERT_SECRET_NAME – Leave it blank to use the GCP-managed certificate.
    - RWO_STORAGE_CLASS=`standard-rwo` OR `premium-rwo` as required.
    - RWX_STORAGE_CLASS=custom storage class name you have created previously for RWX under the filestore csi driver. (e.g.: `custom-rwx`)
6. Run the script in the Helm README for installing DevOps Loop on GKE.

    If the installation fails due to a timeout, run the installation script again.

Results

You have installed DevOps Loop on Google Kubernetes Engine (GKE).

What to do next

You can add users and manage user access by logging in to the Keycloak instance that is installed with DevOps Loop. See User access and administration using Keycloak on page 81.

Installing DevOps Loop on a RHEL system for a demo setup

Disclaimer: In this demo setup, DevOps Loop is deployed on a RHEL-based system with minimum hardware and software specifications. This demo setup is intended solely for evaluation purposes and might not accurately represent the performance, reliability, or feature set of the full production environment. The demo setup might not

include enhanced security configurations typical of production systems. No warranty is implied regarding uptime, support, or continuity of access.

You can find information about the tasks that you can perform to install HCL DevOps Loop on a Red Hat Enterprise Linux (RHEL) system for a demo setup.

Before you begin

You must have completed the following tasks before running the infrastructure setup script:

- Verifed that the RHEL system on which you are deploying DevOps Loop meets the following minimum requirements:
  - Operating system: RHEL 9.4 or later
  - Disk space: 300 GB
  - RAM: 64 GB
  - CPU: 16 Core
- Registered your RHEL system by using the subscription manager.

```
subscription-manager register
```

- (Recommended) Created a snapshot of your VM before running the infrastructure setup script.
- Obtained the user credentials of the licensed Docker account.

Note: If you do not provide the licensed Docker account credentials, the system might take approximately 6 hours to become operational.

1. Run the following command to download and extract the DevOps Loop Helm chart and scripts:

```
helm pull oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201 --untar
```

2. Set the following environment variables:

```
export USERNAME=<docker_username>
export PASSWORD=<docker_password>
```

Note: The `<docker_username>` and `<docker_password>` are your Docker Hub credentials. If these values are not set, you are prompted to log in during script execution.

3. Run the following `RHEL9-K8S-Infra-Setup.sh` script:

```
hcl-devops-loop/scripts/demo/RHEL9-K8S-Infra-Setup.sh 2>&1 | tee infra-setup.log
```

4. Run the following command to create a namespace:

```
kubectl create namespace devops-loop
```

5. Perform the following steps to access the HCL Harbor container registry:
   a. Log in to the HCL Harbor container registry using your HCL ID and password by selecting the **LOGIN VIA OIDC PROVIDER** authentication method.
   b. Copy the **CLI secret** by opening your **User Profile** from the top right corner of the page.

This value serves as the password for the `docker-registry` secret creation command.

c. Create an `imagePullSecret` to authenticate and pull images from the HCL Entitled Registry.

Note: Secrets are namespace-scoped and must be created in each namespace where you plan to install DevOps Loop.

Use the following command to create an `imagePullSecret` named `hcl-entitlement-key`:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --docker-username=<username> \
    --docker-password=<password> \
    --docker-server=hclcr.io
    --namespace devops-loop \
```

For example:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --docker-username=xxxx@hcl.com \
    --docker-password=AOve4CzW2Hs0yCnuQxxxxxxvX \
    --docker-server=hclcr.io
    --namespace devops-loop \
```

6. Update the following values in the `HCL-devops-Install.sh` script:

```
LICENSE_SERVER=<port>@<hostname_or_IP>
EMAIL_SERVER_HOST=<smtp_host>
EMAIL_SERVER_PORT=<smtp_port>
EMAIL_FROM_ADDRESS=<sender_email>
EMAIL_SERVER_USERNAME="<smtp_username>
EMAIL_SERVER_PASSWORD="<smtp_password>
```

7. Run the installation script:

```
./HCL-devops-Install.sh
```

After the successful installation, the following URLs are displayed:

Application URL: `https://<system_IP_address>.nip.io`

Keycloak URL to add users: `https://<system_IP_address>.nip.io/auth`

Note: If you are installing DevOps Loop on a computer that is newly set up, then the installation might fail sometimes. If the installation fails, to resolve the issue, you must uninstall the application by running the `uninstall.sh` script and then reinstall it.

Note: If the installation fails due to a timeout, you must run the installation script again.

8. **Optional:** Run the following command to check whether all the pods are running:

```
kubectl get pods -n devops-loop
```

9. Open a browser and enter the application URL to get started with DevOps Loop.

10. Run the following command to log out of Docker to remove any cached authentication tokens, after completing the DevOps Loop installation:

    ```
    docker logout
    ```

    This command removes the login credentials for `https://index.docker.io/v1/` and cleans up Docker credentials stored in the cache (for example, `/root/.docker/config.json`). Logging out is recommended for security, as Docker authentication tokens are otherwise retained on the system.

    Note:

    - When you run DevOps Loop on a RHEL-based system for an extended period (approximately 30 days), you might need to restart the Calico DaemonSet to ensure continued creation of new loops and Kubernetes pods (for example, when running tests in Test Hub).

      This behaviour is due to Kubernetes security tokens used by the Calico networking component are rotated periodically. In some RHEL demo environments, the updated token is not automatically picked up, and a restart ensures the networking components recognize the refreshed credentials.

      To perform this maintenance step, you must run the following command:

      ```
      kubectl rollout restart daemonset calico-node -n kube-system
      ```

      Restarting the Calico DaemonSet is safe and ensures normal system operation during extended demo periods.

    - Use the following script to recover a Kubernetes node that is experiencing full disk usage:

      a. Locate the recovery script in the repository:

      `ibm-devops-loop/scripts/demo/Recover-K8s-Node.sh`

      b. Run the script on the affected Kubernetes node.

    This script cleans logs, caches, and unused containers to free up disk space and restore node functionality.

Results

You have installed DevOps Loop on a RHEL system for a demo setup.

What to do next

You can onboard users to DevOps Loop. See User access and administration using Keycloak on page 81. After the user is onboarded, you can add the user to teamspaces and loops.

Installation of DevOps Loop on Amazon Web Services (AWS)

You can find information about tasks that you can perform to install DevOps Loop on AWS by using the Elastic Kubernetes Service (EKS).

Preparing AWS infrastructure for installing DevOps Loop on AWS

You must set up the required AWS infrastructure, including Amazon Elastic Kubernetes Service (EKS), storage, and load balancers, before installing DevOps Loop.

Prerequisites

You must have ensured that the following tasks are completed:

- Installed the following AWS tools on the deployment machine:
  - AWS CLI
  - kubectl
  - Helm v3.10 or later
  - eksctl
- Verified cluster access by running the following command:

  ```
  kubectl get nodes
  ```
- Created a VPC with public and private subnets in at least 2 availability zones. Recommended CIDR for example, `VPC CIDR: 10.x.x.x/22`

  Subnet usage:
  - Public subnets: Used for Application Load Balancer (ALB)
  - Private subnets: Used for worker nodes
  - Private subnets: Used for EFS mount targets
- Tagged the subnets correctly for Kubernetes and AWS Load Balancer Controller by using the following values:
  - **Public subnets**

    ```
    kubernetes.io/cluster/<cluster-name> = shared
    kubernetes.io/role/elb = 1
    ```
  - **Private subnets**

    ```
    kubernetes.io/cluster/<cluster-name> = shared
    kubernetes.io/role/internal-elb = 1
    ```

    Note: These tags allow the AWS Load Balancer Controller to automatically discover the correct subnets when creating Application Load Balancers (ALB) and Network Load Balancers (NLB). Without these tags, the load balancer provisioning fails.

About this task

You must perform these AWS infrastructure tasks in the following sequence before installing DevOps Loop.

1. Set up the Amazon EKS cluster on page 64
2. Configure storage (EBS and EFS) on page 64
3. Install the AWS Load Balancer Controller on page 66

4. Configure DNS and TLS on page 67
5. Configure ingress and load balancing on page 67
6. Configure DNS records on page 68

Set up the Amazon EKS cluster

1. Create an Amazon EKS cluster with the following recommended configuration:
    - Node capacity: 8 vCPU, 32 GB memory
    - Multiple worker nodes for high availability
2. Create an **IAM role** for worker nodes and attach the following AWS managed policies:
    - `AmazonEKSWorkerNodePolicy`
    - `AmazonEC2ContainerRegistryReadOnly`
    - `AmazonEKS_CNI_Policy`
3. Create a **security group** to allow traffic within the VPC for communication between:
    - Load balancers
    - Worker nodes
    - Internal Kubernetes services
    - Storage (EFS, EBS)

    Note: Ensure that you deploy worker nodes in private subnets, and use public subnets only for internet-facing load balancers.

4. Enable IAM roles for Service Accounts (IRSA) by enabling the OpenID Connect (OIDC) provider for the cluster by running the following command:

```
eksctl utils associate-iam-oidc-provider \
  --cluster <cluster-name> \
  --approve
```

Note: IRSA is required for components such as the **EFS CSI Driver**, **EBS CSI Driver**, and **AWS Load Balancer Controller**.

Configure storage (EBS and EFS)

1. Perform the following steps to install the Amazon EBS CSI Driver (RWO Storage):
    a. Create an IAM role for EBS CSI Driver:
        - Attach the AWS managed policy: `AmazonEBSCSIDriverPolicy`.
        - Trust Relationship: The role must trust the EKS OIDC provider and allow the service account: `kube-system:ebs-csi-controller-sa`.
    b. Enable the **EBS CSI Driver Add-on**.

During installation, select the IAM role created in the previous step and enable the add-on. This action installs the CSI controller in the `kube-system namespace`.

c. Verify the installation by running the following command:

```
kubectl get pods -n kube-system | grep ebs
```

d. Configure the default StorageClass:
The cluster uses the `gp3 storage class` backed by Amazon EBS.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gp3
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.aws.com
parameters:
  type: gp3
  fsType: ext4
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
```

This storage class is used for workloads requiring ReadWriteOnce (RWO) volumes.

2. Perform the following steps to create an Amazon EFS for shared storage:

a. Create an EFS filesystem by selecting the same VPC used by EKS.

b. Configure mount targets in all Availability Zones used by worker nodes.

c. Create one mount target per Availability Zone where nodes run: AZ-1 → mount target , AZ-2 → mount target, AZ-3 → mount target.

d. Configure security group - Attach a security group to the EFS mount targets as follows. This allows worker nodes to mount the filesystem.

```
Allow NFS access -  Port: 2049 Protocol: TCP Source: EKS node security group
```

3. Perform the following steps to install the EFS CSI Driver (RWX storage):

a. Create an IAM role for EFS CSI Driver (RWX storage)
   - Attach the AWS managed policy: `AmazonEFSCSIDriverPolicy`.
   - Trust Relationship: The role must trust the EKS OIDC provider and allow Kubernetes the service account: `kube-system:efs-csi-controller-sa`.

b. Enable the **EFS CSI Driver Add-on** from the Amazon EFS CSI Driver console and select the **IAM role** created for the EFS CSI driver in the Add-on.

c. Verify the installation by running the following command:

```
kubectl get pods -n kube-system | grep efs
```

d. Create a StorageClass.

Note: Reference the EFS filesystem ID in the EFS StorageClass to enable dynamic provisioning using the EFS CSI driver. For example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap
  fileSystemId: <EFS-FILESYSTEM-ID>
  directoryPerms: "700"
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

e. Verify the storage classes by running the following command:

```
kubectl get storageclass
```

Expected output: `gp3 & efs-sc`

f. Confirm the CSI driver pods by running the following command:

```
kubectl get pods -n kube-system | grep csi
```

Install the AWS Load Balancer Controller

Perform the following steps to install the AWS Load Balancer Controller:

a. Create an IAM policy.

```
AWSLoadBalancerControllerIAMPolicy
```

b. Create an IAM role associated with the service account:

```
kube-system:aws-load-balancer-controller
```

c. Install the AWS Load Balancer Controller via Helm:

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=<cluster-name> \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

This controller automatically provisions AWS load balancers based on Kubernetes resources.

Configure DNS and TLS

1. Create domains by using AWS services

   Note: Use **Amazon Route 53** and **AWS Certificate Manager (ACM)** to configure domains.

2. Create and configure the following hosted zones in Route 53:
   - A public hosted zone (for example, `looptest.xxxx.com`)

   This hosted zone is used as the application domain and enables routing through the Application Load Balancer (ALB).
   - A private hosted zone (for example, `service-looptest.xxxx.com`)

   This hosted zone is used to route traffic to the Network Load Balancer (NLB) for the Loop application.

   Note: Ensure that the domain name includes the `service-` prefix. Make a note of the domain names, as they are required during DevOps Loop installation.

3. Request a certificate using the AWS Certificate Manager (ACM) for public domain (for example: `looptest.xxxx.com`).

   After creating the certificate, make a note of the **Certificate ARN**, as it is required for configuring the Application Load Balancer (ALB).

Configure ingress and load balancing

1. Run the following command to download and run the AWS infrastructure setup script:

```
(TMP_DIR=$(mktemp -d) && \
helm pull hclsoftware/ibm-devops-loop --untar --untardir "$TMP_DIR" --version 2.0.200 > /dev/null 2>&1
 && \
sh "$TMP_DIR/ibm-devops-loop/scripts/AWS/AWS_Emissary_LB_TLS_infra_setup.sh" && \
rm -rf "$TMP_DIR")
```

   Note: This command downloads the DevOps Loop Helm chart, extracts it to a temporary directory, runs the infrastructure setup script, and removes the temporary files after execution.

2. Verify the actions performed by the script.

   The `AWS_Emissary_LB_TLS_infra_setup.sh` script performs the following actions:
   - Creates the **Emissary namespace**
   - Installs **Emissary Ingress** using Helm
   - Creates an **Application Load Balancer (ALB)** for HTTPS traffic
   - Creates an **internal Network Load Balancer (NLB)** with access restricted to the VPC CIDR range for TCP services
   - Configures the required Kubernetes **Ingress resources and Services**

   After running the script, AWS might take 5 - 15 minutes to provision the ALB and NLB. You can verify the creation of ingress resources and services by running the following command:

```
kubectl get ingress -n emissary-system
kubectl get svc -n emissary-system
```

Configure DNS records

1. Obtain the DNS names of the Application Load Balancer (ALB) and Network Load Balancer (NLB) after running the infrastructure script.
2. Create **A records** in **Amazon Route 53** to map your domains:
    - Map the application domain (for example: `looptest.xxxx.com`) to the ALB DNS name.
    - Map the service domain (for example: `service-looptest.xxxx.com`) to the NLB DNS name.

Results

After completing these steps, the required AWS infrastructure is ready, including the EKS cluster, storage, load balancers, and DNS configuration.

What to do next

You must now install DevOps Loop. See Installing DevOps Loop on AWS on page 68.

Installing DevOps Loop on AWS

After you have prepared the infrastructure, you can install DevOps Loop on a newly created EKS cluster by using the Helm chart.

Before you begin

You must have completed the following tasks:

- Gained access to the cluster and installed the following tools:
    - AWS CLI
    - kubectl
    - eksctl
- Installed Helm on the system from which you access the Kubernetes cluster. For more information, refer to Installing Helm.
- Set up an EKS cluster on AWS Elastic Kubernetes Service. For more information, refer to Create an Amazon EKS cluster.
- Prepared the required infrastructure in AWS. See Preparing AWS infrastructure for installing DevOps Loop on AWS on page 63.

1. Log in to the **My IBM Container Software Library** using the ID and password associated with your entitled software.
2. Navigate to the **Entitlement keys** section and select **Copy key** to copy your IBM Entitled Registry entitlement key to your clipboard.
3. Create a secret key in the target namespace `ibm-entitlement-key` to pull images from the IBM Entitled Registry:

```
kubectl create secret docker-registry ibm-entitlement-key \
    --namespace [namespace_name] \
    --docker-username=cp \
```

```
    --docker-password=<EntitlementKey> \
    --docker-server=cp.icr.io
```

4. Run the following command to add the Helm repository:

```
helm pull oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201 --untar
```

5. Run the following command to view the `README.md` file:

```
cat hcl-devops-loop/README.md
```

6. Update the following parameters in the Helm installation script with the appropriate values:
   - `DOMAIN` the FQDN created in Cloud DNS.
   - `TLS_CERT_SECRET_NAME` Leave it blank to use the AWS-managed certificate.
   - `RWO_STORAGE_CLASS=gp2` OR `gp3` as created in the pre-requisite steps.
   - `RWX_STORAGE_CLASS=efs-sc` (The EFS storage class created for RWX access).
7. Run the script in the Helm README for installing DevOps Loop on AWS.

Note: If the installation fails due to a timeout, run the installation script again.

Results

You have installed DevOps Loop on AWS EKS.

What to do next

You can add users and manage user access by logging in to the Keycloak instance that is installed with DevOps Loop. For more details, see User access and administration using Keycloak.

Installation of DevOps Loop in an air-gapped environment

An air-gapped environment is a network or system isolated from the public internet, typically for security or compliance reasons. In such environments, all external resources must be imported manually. DevOps Loop supports air-gapped installations by providing scripts and configurations to transfer required resources into your private environment.

Some deployments operate without internet access, requiring all container images, plugins, and dependencies to be preloaded into a private registry.

Scripts and configurations for Air-gapped installation

DevOps Loop provides the `airgap.sh` script along with Helm configuration parameters to support air-gapped setups.

Parameter	Description	Example value
`global.imageRegistry`	Private registry for product images	`registry.local:5000/devops-loop`
`global.externalImageRegistry`	Private registry for unqualified/public images. The `airgap.sh` script places qualified and unqualified images in the same image repository, you can separate them if required.	`registry.local:5000/public`

Parameter	Description	Example value
`airgap.sh`	Script to list or copy required images to private registry	`./airgap.sh -|`

The `airgap.sh` script provides the following capabilities:

- Lists all the container images required by DevOps Loop (using `-l` option).
- Copies images from public registries to a private or internal registry.

Learning Loop in an air-gapped environment

Learning Loop sample applications use npm to install JavaScript dependencies during the build process. In an air-gapped environment, the public npm registry is unavailable. As a result, build steps such as `npm ci` fail unless the required npm packages are made available through an internal npm registry.

Before disconnecting the environment from the internet, populate an internal npm registry with all packages required by the Learning Loop sample applications and update the application lockfiles to reference the internal registry. After this preparation is complete, Learning Loop builds can run entirely within the air-gapped environment. For more information, see Preparing Learning Loop applications for an air-gapped environment on page 74.

Note: This preparation is required only for Learning Loop applications that use npm packages. Standard DevOps Loop installation images are handled separately by the `airgap.sh` script.

Preparing images and resources for an air-gapped installation

Before you begin the air-gapped installation of DevOps Loop, you must prepare all the required container images and resources. In an air-gapped environment, external repositories are not accessible, and therefore, you must collect and transfer all required assets to your internal registry or offline storage.

Before you begin

You must have completed the following tasks:

- Gained administrative privileges on the target Kubernetes namespace and the airgap container registry.
- Downloaded the DevOps Loop Helm chart and it is accessible.
- Downloaded the `airgap.sh` script and ensured that it is available in the DevOps Loop Helm chart.
- Installed the `skopeo` utility on your local machine and authenticated with the required image registries.

- Allocated sufficient disk space to save product images and plugin files for transfer to the air-gapped environment.

Note: Product images consume approximately 50 GB or more of disk space in your internal/air-gapped image registry.

About this task

You must perform the following tasks in sequence to prepare for an air-gapped installation:

1. Copy DevOps Loop images by using a jump server on page 71 OR

   Copy DevOps Loop images by using file-based transfer on page 71
2. Prepare MongoDB for an air-gapped installation on page 72
3. Prepare MySQL JDBC driver on page 72

Copy DevOps Loop images by using a jump server

Copy DevOps Loop images directly to the internal registry by using a jump server. You can use this method when a system can connect to both the public internet and the air-gapped image registry.

Run the following command to copy the images directly to the internal registry:

```
sh ./hcl-devops-loop/scripts/airgap/airgap.sh -r hcl-devops-loop -d registry.local:5000/devops-loop
```

Results

DevOps Loop images are copied directly to the internal image registry.

OR

Copy DevOps Loop images by using file-based transfer

Copy DevOps Loop images to an air-gapped environment by using file-based image archives. You can use this method when no system has access to both the public internet and the air-gapped environment.

1. Run the following command on a machine with internet access:

   ```
   sh hcl-devops-loop/scripts/airgap/airgap.sh -r hcl-devops-loop -d loopimagesdir -x save -y
   ```
2. Transfer the `loopimagesdir` directory to a machine inside the air-gapped environment.
3. Run the following command to load the images into the internal registry:

   ```
   sh ibm-devops-loop/scripts/airgap/airgap.sh -r ibm-devops-loop -d airgap.registry.io/loopimages -s
    loopimagesdir -x load
   ```

Results

DevOps Loop images are available in the internal air-gapped image registry.

Prepare MongoDB for an air-gapped installation

You must prepare MongoDB for air-gapped installation by making the required Helm chart available locally and updating the configuration to use images from your internal registry.

1. Download and extract the MongoDB Helm chart version 14.13.0 from Bitnami Helm Charts into the same directory as the expanded `ibm-devops-loop` Helm chart.
2. Update the installation script as follows:
   - Set `MONGO_IMAGE_REPO=registry.local:5000/devops-loop`.
   - Replace `bitnami/mongodb` with `mongodb` in the `helm upgrade --install` command.

Prepare MySQL JDBC driver

The DevOps Deploy server, installed as part of DevOps Loop, creates a MySQL database to store various artifacts. In an air-gapped environment, the installation cannot automatically download the required JDBC driver. You must make the driver available manually before starting the DevOps Loop installation.

1. Download the required MySQL JDBC driver and make it accessible within the air-gapped environment.
2. Write a bash script, named `script.sh` that copies the JDBC driver from a location accessible from your cluster to `${UCD_HOME}/ext_lib/`.
3. Store the script `script.sh` in a YAML file describing a Kubernetes ConfigMap.
   The Kubernetes ConfigMap is defined for copying the JDBC driver to your Persistent Volume (PV) during the chart installation process.
   Here is an example ConfigMap YAML file that copies a MySQL .jar file from a web server using `wget`:

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: user-script
   data:
     script.sh: |
       #!/bin/bash
       echo "Running script.sh..."
       if [ ! -f ${UCD_HOME}/ext_lib/mysql-jdbc.jar ] ; then
         echo "Copying file(s)..."
         wget -L -O mysql-jdbc.jar http://webserver-example/mysql-jdbc.jar
         mv mysql-jdbc.jar ${UCD_HOME}/ext_lib/
         echo "Done copying."
       else
         echo "File ${UCD_HOME}/ext_lib/mysql-jdbc.jar already exists."
       fi
   ```

4. Create the ConfigMap in your cluster by running the following command:

   ```
   oc create configmap <map-name> <data-source>
   ```

5. Reference the ConfigMap name in Helm by using the following command:

   ```
   hcl-launch-server-prod.extLibVolume.configMapName
   ```

Results

You have copied all required DevOps Loop images and prepared the necessary resources, including MongoDB and the MySQL JDBC driver. Your internal registry or offline storage now contains all required artifacts.

What to do next

You can begin the air-gapped installation of DevOps Loop. See Installing DevOps Loop in an air-gapped environment on page 73.

Installing DevOps Loop in an air-gapped environment

After preparing the required images, plugins, and JDBC drivers in your air-gapped registry or offline storage, you can begin installing DevOps Loop in an air-gapped environment by using Helm.

Before you begin

You must have completed the following tasks:

- Preloaded images and resources. See Preparing images and resources for an air-gapped installation on page 70.
- Ensured that you have administrative privileges on the Kubernetes namespace where DevOps Loop will be installed.
- Installed and configured Helm on your system.
- Gained access to the airgap image containing the preloaded images.

1. Modify the `ADDIITIONAL_HELM_OPTIONS` variable in the DevOps Loop installation script, to set the following Helm values to point to your airgap registry:

```
--set global.imageRegistry=registry.local:5000/devops-loop
--set global.externalImageRegistry=registry.local:5000/devops-loop
--set platform.velocity.plugins.plan.image=ucv-ext-compass:1.2.2
--set platform.velocity.plugins.control.image=hclcr.io/accelerate/ucv-ext-control:1.1.1
--set platform.velocity.plugins.test.image=hclcr.io/accelerate/ucv-ext-onetest-server:1.0.32
--set platform.velocity.plugins.deploy.image=hclcr.io/accelerate/ucv-ext-launch:5.1.1
--set platform.velocity.plugins.build.image=hclcr.io/accelerate/ucv-ext-build:1.1.2
--set platform.velocity.plugins.yaml.image=hclcr.io/accelerate/ucv-ext-yaml-executor:0.0.13
```

2. Edit the DevOps Loop installation script to use the locally downloaded and expanded DevOps Loop Helm chart.
   For **OpenShift (OCP)** installations, no further changes to the installation script are required.
   For **Kubernetes (K8s)** installations, you must locate the `helm upgrade --install` command at the end of the installation script and replace `ibm-helm/ibm-devops-loop` with `ibm-devops-loop`.
3. Follow the DevOps Loop installation instructions based on your environment.

Results

You have installed DevOps Loop in the air-gapped environment.

What to do next

Before installing additional plug-ins, ensure the required images are available in your internal or air-gapped environment:

- If you want to install Measure plug-in images that are not included in the standard DevOps Loop set, download them from their public repositories (such as docker.io or hclrc.io) and upload them to your internal or air-gapped image registry.
- If you want to install Deploy plug-ins that are not included in the standard set, download them from their public repositories to local storage, then upload them to the DevOps Deploy server.

Related reference

Installation of DevOps Loop on page 45

Preparing Learning Loop applications for an air-gapped environment

Before running Learning Loop applications in an air-gapped environment, prepare an internal npm registry with the required npm packages and update the application lockfiles to use the internal registry. This preparation enables `npm ci` and subsequent builds to complete without accessing the public npm registry.

Before you begin:

Ensure that you have completed the following tasks:

- Installed DevOps Loop in an air-gapped environment. See Installing DevOps Loop in an air-gapped environment on page 73.
- Configured an internal npm registry that is accessible from the DevOps Loop build environment.
- Verified that the environment has temporary internet access to download npm packages.
- Installed npm client.
- Installed and configured `kubectl` with access to the DevOps Loop namespace.

Procedure for environments with temporary internet access

1. Clone the Learning loop frontend and backend repositories to a machine that has internet access.

   ```
   git clone <frontend-repository-url> ${LOOP_NAME}-frontend
   git clone <backend-repository-url> ${LOOP_NAME}-backend
   ```

2. Install and configure an internal npm registry that can be reached by the DevOps Loop build agents.

   For example, to install Verdaccio:

   ```
   sudo npm install -g verdaccio
   ```

   Configure the registry to proxy requests to the public npm registry while internet access is available, and then start the service.

3. Copy the frontend and backend repositories to the DevOps Loop build pod:

   ```
   kubectl -n devops-loop cp "${FRONTEND_REPO_PATH}/." \
   devops-build-agent-0:/tmp/${LOOP_NAME}-frontend
   ```

   ```
   kubectl -n devops-loop cp "${BACKEND_REPO_PATH}/." \
   devops-build-agent-0:/tmp/${LOOP_NAME}-backend
   ```

4. Connect to the build pod and configure npm to use the internal registry:

```
npm config set registry "${REGISTRY_URL}"
npm config set replace-registry-host always
npm config set omit-lockfile-registry-resolved true
```

5. Regenerate the `package-lock.json` file for each repository.

```
rm -f package-lock.json
npm install --package-lock-only --ignore-scripts
```

   Verify that the regenerated lockfiles do not reference the public npm registry.

6. Run the following command for each Learning Loop repository while internet access is still available:

```
npm ci
```

   This command downloads all required dependencies and stores them in the internal npm registry for later use.

7. Copy the regenerated `package-lock.json` files back to the cloned repositories. Commit the updated lockfiles to a feature branch and push the branch to your source control repository.

   For example:

```
git checkout -b chore/internal-registry-lockfile
git add package-lock.json
git commit -m "Regenerate package-lock.json for internal npm registry"
git push -u origin chore/internal-registry-lockfile
```

   Create and merge a pull request for both the frontend and backend repositories.

8. Remove any proxy or uplink configuration from the internal npm registry, after all required npm packages have been downloaded.
9. Disconnect the environment from the internet.
10. Run the Learning Loop build.

    Verify that the build completes successfully and that npm ci retrieves all required packages from the internal npm registry without attempting to access the public npm registry.

Procedure for fully air-gapped environments

If the target environment does not have internet access at any stage, prepare the required npm packages on a connected staging machine before deploying them to the air-gapped environment.

1. Install and configure an internal npm registry on a staging machine with internet access, .
2. Configure the registry to proxy requests to the public npm registry.
3. Prepare the Learning Loop application manifest files (`package.json` and `package-lock.json`).
4. Regenerate the `package-lock.json` files and run `npm ci` for the frontend and backend applications to populate the internal npm registry with all required npm packages.
5. Remove any proxy or uplink configuration from the internal npm registry so that it operates in offline mode.
6. Export the registry data and the application manifest files.

7. Transfer the exported registry data and application manifest files to the air-gapped environment by using your organization's approved transfer method.
8. Import the registry data into the internal npm registry in the air-gapped environment.
9. Configure the DevOps Loop build environment to use the internal npm registry.
10. Run a Learning Loop build and verify that `npm ci` installs the required packages from the internal npm registry without requiring internet access.

The Learning Loop applications are configured to use the internal npm registry, and builds can run successfully in the air-gapped environment without requiring access to the public npm registry.

Managing DevOps Loop features after installation

After you install HCL DevOps Loop, you can manage and customize certain features based on your requirements.

During the installation of DevOps Loop, if you did not use any Helm parameters, you can still use those parameters even after installation is complete.

You can use the helm upgrade command to enable or disable server features after the installation of DevOps Loop is complete.

The sample code is to enable the Egress policy that restricts traffic to private IP addresses by retaining the values of other parameters that you used during the installation.

Similarly, you can use the other additional Helm parameters to enable or disable features based on your requirements. See Additional Helm parameters on page 76.

Additional Helm parameters

You can find the information about Helm parameters that you can use during the installation of HCL DevOps Loop.

As a system administrator, you must complete the additional configuration for the capabilities by using Helm parameters. Each Helm parameter that is specific to the solution is prefixed by its Helm chart name as follows:

- Plan: `hcl-devopsplan-prod`
- Test: `hcl-devops`
- Deploy: `hcl-launch-server-prod`
- Measure: `hcl-velocity`

For example, to configure a property in Plan, use the following command:

```
bash
--set hcl-devopsplan-prod.property=value
```

You can run the following commands to view the required parameters:

- Plan: helm show readme hcl-helm/hcl-devopsplan-prod # DevOps Plan
- Deploy: helm show readme hcl-helm/hcl-launch-server-prod # DevOps Deploy

- Test: helm show readme hcl-helm/hcl-devops # DevOps Test
- Measure and Release: helm show readme hcl-helm/hcl-velocity # DevOps Velocity

License management and user administration

After you install DevOps Loop, you must manage licenses, users, and authentication to control access to platform resources.

License management

During installation, you can configure licensing information by using the license server settings that are provided in the Helm chart. For more information, see DevOps Loop licensing information on page 78.

The **User Administration** page displays license consumption information for the tenant. Administrators can view license usage across configured license tiers and manage user license assignments.

The **Licenses** section displays:

- The available license tiers.
- The total number of licenses in each tier.
- The number of licenses that are currently assigned.
- The number of licenses that remain available.

License usage information is updated automatically when user license assignments are modified.

User administration

As part of administration, you can add users to DevOps Loop and assign roles in Keycloak. For more information, see User access and administration using Keycloak on page 81 and Enabling the social sign-up and social login for DevOps Loop on page 80.

You can use the default user management capabilities that are provided by the platform or manage users through the configured identity provider.

An administrator can manage users within a tenant. A tenant represents an organization within which multiple TeamSpaces can be created, and each TeamSpace can contain multiple Loops.

To access user administration, click **Settings** > **User Administration**.

Tenant information

The top section of the **User Administration** page displays summary information about the tenant, including:

- Tenant name
- Tenant ID
- Creation date
- Number of TeamSpaces

- Number of Loops
- Number of users

Managing users

The user list displays all users that belong to the tenant. You can perform the following actions:

- Browse users within the tenant.
- Search for users by name or email address.
- Filter users by TeamSpace.
- Filter users by Loop.
- Assign or modify a user's license tier.

Select a user from the list to view detailed user information.

User details

When you select a user, the details pane displays information about the user, including:

- User name
- Email address
- Assigned role
- Associated TeamSpaces
- Associated Loops

This information helps administrators review user access and license assignments across the tenant.

Roles

User roles are managed through Keycloak and are reflected in DevOps Loop. The assigned role for a user is displayed in the user details pane.

For information about managing users and roles in Keycloak, see User access and administration using Keycloak on page 81.

DevOps Loop licensing information

You can find the details of licensing that HCL DevOps Loop supports.

DevOps Loop supports the following 3 tiers of license:

- **Essentials** (Tier 1): Plan, Control and Code
- **Standard** (Tier 2): Plan, Control, Code, Build, Test, Deploy and Harbor (if enabled).
- **Premium** (Tier 3): Plan, Control, Code, Test, Build, Deploy, Measure, Release and Harbor (if enabled).

Each tier is defined by the number of capabilities that are included in the DevOps Loop platform.

Licenses are consumed when the admin assigns them, and only released when the admin explicitly revokes them or DevOps Loop is uninstalled. License tier re-assignments may take up to 15 minutes to reflect.

By default, non-admin users must be assigned a tier before logging in, while admin users automatically receive the highest available tier. This per-user tier assignment allows different access levels to be enforced for different users, providing flexibility in license management.

Configuration

When you purchase one of these tiers, you must map the licenses to My HCLSoftware (MHS). MHS is a cloud-based web application that helps to manage software entitlements and licenses. When you install DevOps Loop, you must specify the License server URL and ID, which are configured only through the Helm chart. For more information about managing licenses for your product, refer to the following resources:

- My HCLSoftware
- Managing HCL Local License Server or Managing HCL Cloud License Server

License allocation

You must run the following command to retrieve information on the licensing allocation:

curl --cacert /path/to/your/certificate.crt https://FQDNhostname:LLSport/v1/licensepools/poolID

Sample output:

```
{
  "id": "MyPoolID",
  "features": {
    "DevOpsAutomationEssentials": {
      "total": 100,
      "leased": 0,
      "available": 100,
      "name": "DevOps Automation Essentials User"
      }
    }
}
```

When you run the command to retrieve the current license allocations, depending on the tier that you have opted for, the following values are displayed:

- **DevOpsAutomationEssentials**
- **DevOpsAutomationStandard**
- **DevOpsAutomationPremium**

The license feature names that are used by the License server are different from the product name.

Usage

After the Admin user creates a user, when the user logs in to DevOps Loop, then one license of the highest available tier is consumed. The user can access the products that are included in the licensing tier that is purchased.

DevOps Loop provides a user interface to view and manage licensing. You can revoke or assign a user's license through the **User Administration** page. See License management and user administration on page 77 and Revoking a license on page 81.

Troubleshooting

You can view the licensing logs for troubleshooting by running the following command:

```
kubectl logs <licensing_pod_name> -n <namespace>
```

Where, the *namespace* is defined by the Helm installation. You can find the *licensing_pod_name* by running the following command:

```
kubectl get pods -n <namespace>
```

The pod name that contains 'licensing' in the name is listed.

Enabling the social sign-up and social login for DevOps Loop

You can enable social authentication for users to sign up and log in to HCL DevOps Loop by using a social network account. The Keycloak admin can configure this social sign-up in the Keycloak UI.

Before you begin

You must have completed the following tasks:

- Installed DevOps Loop.
- Assigned the role of a Keycloak Administrator.
- Copied the URL of the Keycloak UI, username, and password on completion of the installation of the platform.
- Obtained the credentials to access the dashboard of a social identity provider to configure the delegation of authentication to a social media account.
- Read and understood Integrating identity providers.

About this task

The social sign-up link on the **Login** page of DevOps Loop is not visible when you install DevOps Loop by using the default values. The sign-up link is hidden to provide enhanced security to the platform. You can enable the sign-up link on the **Login** page when you want to provide self-registration of users in DevOps Loop.

The following procedure describes steps to be performed to enable the **Sign up** link by using Keycloak UI.

1. Log in to the Keycloak Admin Console.
2. Click **Identity providers** from the left navigation pane.
3. Select an identity provider under the **Social** section.
   The configuration page for the selected identity provider is displayed.
4. Enter the configuration details.
   The fields in the configuration page depend on the social identity provider that you have selected.
5. Click **Add** to apply the changes.

Results

You have enabled the social sign-up link on the **Login** page of DevOps Loop.

What to do next

You must add the self-registered users to the **Users** group in the Keycloak Admin Console. See User access and administration using Keycloak on page 81.

Related information

Keycloak

Revoking a license

Each user is assigned a license when they log in to the DevOps Loop for the first time. As an admin, you can revoke the assigned license if the user is no longer part of the platform.

1. Log in to DevOps Loop.
2. Navigate to **Settings > User Administration**.
   The **User Administration** page and all the users are displayed. You can search for a user by their name or email ID if the list is long.
3. Select **None**from the **Tier** option next to the username.

   A confirmation dialog is displayed.
4. Click **Yes, Revoke**.

Results

You have revoked the assigned license for a user.

User access and administration using Keycloak

HCL DevOps Loop uses Keycloak (https://www.keycloak.org/) to manage and authenticate users. You can manage user access by logging in to the Keycloak instance that is installed with DevOps Loop.

Keycloak uses the concept of a realm to manage and authenticate users. When you install DevOps Loop, a realm called *platform* is created for you in Keycloak. All server users belong to this realm and when they log in to DevOps Loop, they log into that realm.

As an administrator, it is important to consider the following points about the platform administration:

- To begin with, there is no administrator for DevOps Loop.

  Such an administrator is required for accessing additional functions, which include claiming ownership of projects and unarchiving them. However, you can assign administrative privileges to any user. You must assign the privilege by adding the admin role to the user in Keycloak.

- You must add a user that you want to be the administrator in Keycloak by logging in to the Keycloak Admin Console at `https://<fully-qualified-dns-name>/auth/`.

  Note: Do not use that admin user to perform non-administration tasks. Instead, add another user.

  The default username for the Keycloak administrator is `keycloak`. The password is randomly generated when the software is installed. You can see the password by using the following kubectl command:

  ```
  kubectl get secret -n <namespace> <helm name>-keycloak -o jsonpath="{.data.password}" | base64 --decode;
   echo
  ```

- After you add the user that you want to be the administrator for DevOps Loop, you must make that user the administrator.

  In the Keycloak Admin Console, on the **Users** page, you can search and select the user that you want to make an administrator. Then, in the **Groups** tab, you can add the user to the **Admins** group.

- All users must have the **Users** group assigned to them to access DevOps Loop.

  In the Keycloak Admin Console, on the **Users** page, you can search and select a user, and then in the **Groups** tab, you can add the user to the **Users** group.

  For more information about assigning user roles, see Groups in the Keycloak documentation.

Now that you are the platform administrator, it is important to consider the following points about the default user management and authentication:

- Minimum password length is 8 characters
- Email verification of new users is turned off
- The Forgot Password feature is turned on by default but no instructions are sent to the user to reset their password
- Forgotten user passwords are changed by you if you do not enable Keycloak to send instructions to reset a password

Note: If a user is added through the user management of HCL DevOps Plan, then that user must have the **Users** or **Admins** group assigned in the Keycloak Admin Console to get access to DevOps Loop.

You can review the following sections about changing the default authentication controls.

Email settings

The default status of the Forgot Password switch is ON in the *devops-automation* realm. However, as an administrator, you must enable Keycloak to send an email to the user with instructions to reset their password. If you want to verify an email, you must also enable Keycloak to send an email to the user to verify their email address.

You must provide SMTP server settings for Keycloak to send an email. After you log in to the Keycloak Admin Console, see Email Settings in the Keycloak documentation.

Then, to set up the email verification, see Forgot Password in the Keycloak documentation.

Password policy

The *devops-automation* realm has a password policy where the minimum length of a password is 8. As an administrator, you can update password policies in Keycloak.

After you log in to the Keycloak Admin Console, see Password Policies in the Keycloak documentation.

User password

When you create a user, you must create credentials for the user by clicking the **Set password** button in the **Credentials** tab. Then, you must share the username and password with the user. While setting the password, if the **Temporary** slider button is set to **On**, then the user must set a new password before logging in to the platform.

If you did not enable Keycloak to send instructions to a user about how to reset a password, you must use the Keycloak Admin Console to change their password for them.

After you log in to the Keycloak Admin Console, see User Credentials in the Keycloak documentation.

User deletion

When a user is inactive or no longer needs to access the platform, you can delete that user.

After you log in to the Keycloak Admin Console, see Deleting Users in the Keycloak documentation.

About user roles and access permissions

Users have different roles and access levels on the platform and in integrated capabilities, depending on their tasks and the settings in each application.

Two types of roles are supported in HCL DevOps Loop: admin and user. Admin users can create teamspaces and loops.

Role	Permissions
Admin	Admins have full permission for the following features in the platform: • User administration • Integration • Teamspace management • Loop management • Loop Genie • Dashboards
User	Users have view permission for the following features in the platform:

Role	Permissions
	• View loops that the user is a part of • Switch to the application • Loop Genie • Dashboards

Notes:

- Loop creator is granted the highest permissions for the resources in the integrated applications that are created by the loop. The other invited users are granted basic access. The loop creator can elevate the other users' access as required.
- If you are an admin user in the platform but a loop member, in Deploy, you will be given the same privileges as the loop creator. For information on roles and permissions in Deploy, see DevOps Loop: Deploy security management.

Backup and restoration of DevOps Loop

You must back up HCL DevOps Loop data before you uninstall the current version of DevOps Loop. You must back up the data to avoid data loss or inaccessibility.

A backup creates a copy of your data and stores it in secondary storage. You can use this copy to recover if the original data is lost or becomes inaccessible.

A restore copies backed-up data from secondary storage back to its original location, making it available again.

You must back up and restore the data when you perform the following tasks:

- Move the existing environment to a new system.
- Change the name of the release or namespace that you used during the installation of DevOps Loop.
- Minimize the downtime of DevOps Loop during disaster recovery.

Tools supporting backup and restoration

DevOps Loop uses the following open source tools to perform backup and restore operations:

- **Velero**: It is a Kubernetes native backup and restore tool that captures and restores Kubernetes resources, persistent volumes, and application state. Velero also supports disaster recovery and migration scenarios between clusters.
- **MinIO**: It is an S3 compatible object storage system used as the backup storage target for Velero. MiniO stores the backups created by Velero and enables you to retrieve them for restore operations.

Velero is installed in your Kubernetes cluster and configured to use MinIO or any other S3-compatible storage backend.

Note: MinIO is provided here as an example S3-compatible storage solution for reference purposes only.

To install Velero and MiniO in your cluster, you must run the following script:

```
./ibm-devops-loop/scripts/backup-restore/install-minio.sh
./ibm-devops-loop/scripts/backup-restore/install-velero.sh
```

After installing MinIO, run the port-forward commands to make it accessible on your local machine:

```
```bash
kubectl port-forward --address 0.0.0.0 pod/minio -n loops-minio 9000:9000 9090:9090
```

Then, use the MinIO Web UI (http://localhost:9090) to create a bucket (for example, "loops-backup") to store Velero backups.

Backing up DevOps Loop data

You can back up DevOps Loop by using Velero and Minio to support disaster recovery and upgrade scenarios.

Before you begin

You must have completed the following tasks:

- Ensured that all DevOps Loop services are running and in a healthy state.
- Gained administrative privileges.
- Installed Velero and configured with an S3-compatible storage backend (MinIO or AWS S3). See Tools supporting backup and restoration on page 84.
- Locally saved the Helm values for your current DevOps Loop deployment.

About this task

DevOps Loop utilizes both block and file storage, allowing the back up process to support CSI snapshots and fs-volume backups.

"

Important: If DevOps Loop pods that use file storage for fs-volume backups are not labeled, Velero will not include their data in the backup, which could result in incomplete or inconsistent backup data.

"

1. Run the following command to label DevOps Loop pods that use file storage for fs-volume backups:

   ```
   ./ibm-devops-loop/scripts/backup-restore/annotate-pods.sh
   ```
   ```

Note: If your DevOps Loop deployment only uses block storage volumes (CSI snapshots), you can skip this step.

2. Export the Helm configuration of your DevOps Loop deployment to a local file by running the following command:

```
helm get values devops-loop -n devops-loop > devops-loop-values-backup.yaml
```

3. Run a Velero backup of the DevOps Loop namespace, including PVCs, PVs, and Secrets, using the following command:

```
velero backup create devops-loop-backup \
  --include-namespaces devops-loop \
  --include-resources persistentvolumeclaims,persistentvolumes,secrets \
  -n velero
```

4. Run the following command to verify the backup status to ensure it completed successfully:

```
velero backup describe devops-loop-backup --details
```

5. **Optional** Backup any external databases with their native backup tools to ensure data consistency.
6. Optional: Check the status and logs of your restore jobs to troubleshoot any issues:

```
# Check backup status
velero get backup

# Describe specific backup
velero describe backup <backup-name>

# View backup logs
velero backup logs <backup-name>
```

Results

You have created a backup of the DevOps Loop namespace and Helm values for disaster recovery or upgrade scenarios.

Restoring DevOps Loop data

You can restore DevOps Loop from a backup created with Velero and Helm values after a disaster recovery event or a failed upgrade.

Before you begin

You must have completed the following tasks:

- Ensured that the Kubernetes environment hosting DevOps Loop is running and accessible.
- Gained administrative privileges
- Installed Velero and configured with an S3-compatible storage backend (MinIO or AWS S3). See Tools supporting backup and restoration on page 84.
- Locally saved the Helm values for your current DevOps Loop deployment.

About this task

DevOps Loop utilizes both block and file storage, allowing the restore process to support CSI snapshots and fs-volume backups.

“

Important: If DevOps Loop pods that use file storage were not labeled during backup, their data will not be restored, which could result in incomplete or inconsistent recovery.

”

1. Run the following command to restore the DevOps Loop namespace from the Velero backup:

```
velero restore create --from-backup devops-loop-backup
```

2. Verify that the restore is completed successfully by running the following command:

```
velero restore describe devops-loop-backup --details
```

3. Reinstall or upgrade DevOps Loop using the saved Helm values if necessary:

```
helm upgrade --install devops-loop devops-loop-chart -n devops-loop -f devops-loop-values-backup.yaml
```

Note: If persistent volumes are larger than the default Helm chart sizes, override them using `--set` options during reinstall.

4. **Optional**: Restore any external databases used by DevOps Loop using their native restore tools to ensure data consistency.
5. Restart DevOps Loop to ensure all services reflect the restored state:

```
kubectl rollout restart deployment devops-loop -n devops-loop
```

6. Optional: Check the status and logs of your restore jobs to troubleshoot any issues:

```
# Check all restores
velero get restore

# Describe a specific restore
velero describe restore <RESTORE_NAME>

# View restore logs
velero restore logs <RESTORE_NAME>
```

Results

You have restored the DevOps Loop namespace, including all CSI snapshots and fs-volume backups, and any external databases (if applicable).

Private CA and self-signed certificate support

A private Certificate Authority (CA) is an organization-specific service that issues and manages digital certificates for use within an enterprise’s private network. These certificates secure communication between internal applications and services.

By default, self signed or privately signed certificates are not automatically trusted by clients because they are not included in the public trust stores used by operating systems and browsers.

DevOps Loop supports configuring and trusting such private CA or self-signed certificates so that secure connections to internal systems can be established.

During installation or upgrade, DevOps Loop reads CA certificates from a Kubernetes secret specified by the Helm parameter `global.privateCaBundleSecretName`. These certificates are then added to internal trust store of DevOps Loop so that integrations and plugins can establish encrypted, trusted connections to systems that use private CA or self-signed certificates.

You can also reference the same secret using `global.ibmCertSecretName` when using self-signed certificates to simplify configuration.

Related information

Configuring trusted certificates in DevOps Loop on page 88

Configuring trusted certificates in DevOps Loop

You can configure DevOps Loop to use private CA and self-signed certificates by creating or updating a Kubernetes secret with a PEM certificate bundle.

Before you begin

You must have completed the following tasks:

- Ensured that you have the administrator privileges to the Kubernetes namespace where DevOps Loop will be installed.
- Ensured that the private CA is available in a PEM format (`.pem`).

Note: In the scenarios below, `mycacrt.pem` is used as a sample PEM file name for a private CA or combined certificate bundle. You must replace `mycacrt.pem` with the actual name and path of your PEM file.

Perform any of the following actions based on the scenario:

Scenario	Action	Notes
No certificate provided (auto-generate self-signed)	Perform the following steps: a. Set `SELF_SIGNED=true` in the DevOps Loop installation script. The script generates `key.pem` and `cert.pem` valid for 365 days with SAN for `$DOMAIN`.	No manual secret creation is required. The certificate is generated automatically.

Scenario	Action	Notes
	The script creates a Kubernetes secret named in `devops-loop-tls-secret` containing `ca.crt=cert.pem`, `tls.crt=cert.pem`, and `tls.key=key.pem`. The script also sets `global.hclCertSecretName=devops-loop-tls-secret`, so that it is used to terminate TLS for the DevOps Loop instance.	

Using a private CA certificate bundle	Perform the following steps:
	
	a. Combine multiple CA/intermediate certificates if needed:
	`cat rootCA1.pem rootCA2.pem > mycacrt.pem`
	b. Create the secret:
	`kubectl create secret generic privateca-secret --from-file=ca.crt=/path/to/mycacrt.pem -n devops-loop`
	c. Edit the `ADDITIONAL_HELM_OPTIONS` variable in the DevOps Loop installation script to add:
	`--set global.privateCaBundleSecretName=privateca-secret -set hcl-devops-prod.ingress.cert.selfSigned=true`
	
	The helm value
	`hcl-devops-prod.ingress.cert.selfSigned=true`
	is needed to work-around an issue with the latest shipping version of DevOps Test when using TLS certificates signed by a private CA.
		Used when internal services are signed by a private CA. The `ca.crt` key is mandatory.

Updating an existing CA or self-signed certificate	Perform the following steps: a. To prevent the need to manually restart pods, create a new secret with the updated certificate in the secret: `kubectl create secret generic privateca2-secret \ --from-file=ca.crt=/path/to/mycacrt.pem -n devops-loop`	

Scenario	Action	Notes

	b. Edit the `ADDITIONAL_HELM_OPTIONS` variable in the DevOps Loop installation script to add:
	`--set global.privateCaBundleSecretName=privateca2-secret -set hcl-devops-prod.ingress.cert.selfSigned=true`
	The helm value
	`hcl-devops-prod.ingress.cert.selfSigned=true`
	is needed to work around an issue with the latest shipping version of DevOps Test when using TLS certificates signed by a private CA.	

Results

You have configured DevOps Loop to use the specified trusted certificates.

What to do next

You must run the DevOps Loop installation script. See Installation of DevOps Loop on page 45.

Upgrading DevOps Loop

After you complete the required backup and pre-upgrade tasks, you can upgrade DevOps Loop to a newer version.

Before you begin

Ensure that you have completed the following tasks:

- Backed up DevOps Loop. See Backing up DevOps Loop on page 84.
- Ensured that you have access to the Kubernetes cluster and the Helm CLI.
- Verified that the Helm chart repository is reachable from your environment.
- Deleted any existing namespaces that are used by Code dev containers running in user namespaces, when applicable.

About this task

You can upgrade DevOps Loop by using one of the following methods:

- **Manual Helm upgrade** - Use this method to upgrade the DevOps Loop platform without enabling Harbor.
- **Installation script upgrade** - Use this method if you want to enable Harbor during the upgrade. Configure the Harbor parameters before you run the installation script.

Note: Upgrading the DevOps Loop platform updates the platform components only. Existing loops are not modified and continue to use their existing sample data, tasks, and Loop Genie TASKS repository and velocity plugins. To use the sample applications, tasks, and Loop Genie capabilities that are introduced in version 2.0.2, create a new loop after the platform upgrade.

Upgrade DevOps Loop without Harbor

1. Run the following commands to set environment variables for the upgrade:

```
HELM_NAME=devops-loop
NAMESPACE=devops-loop
LOOP_CHART_VERSION=2.0.201
```

2. Run the following command to initialize the additional Helm options:

```
export ADDITIONAL_HELM_OPTIONS="--set option1=value1 --set option2=value2"
```

3. Run the following command to retrieve the existing Helm values:

```
helm get values ${HELM_NAME} \
  --namespace ${NAMESPACE} \
  > user_input_values.yaml
```

4. Retrieve the existing Helm values again to verify the configuration by running the following command:

```
ADDITIONAL_HELM_OPTIONS=${ADDITIONAL_HELM_OPTIONS:-}
helm get values ${HELM_NAME} --namespace ${NAMESPACE}
```

5. Run the following command to upgrade DevOps Loop:

```
helm upgrade "${HELM_NAME}" oci://hclcr.io/devops-automation-helm/hcl-devops-loop \
  --namespace ${NAMESPACE} \
  --version=${LOOP_CHART_VERSION} \
  -f user_input_values.yaml \
  ${ADDITIONAL_HELM_OPTIONS}
```

6. Run the following command to verify that all the DevOps Loop pods and components are running:

```
kubectl get pods -n devops-loop
```

You have upgraded DevOps Loop without enabling Harbor.

Upgrade DevOps Loop with Harbor enabled

1. Login to the HCL container registry:

```
helm registry login hclcr.io \
  --username <username> \
  --password <password>
```

2. Run the following command to download the Helm chart:

```
helm pull oci://hclcr.io/devops-automation-helm/hcl-devops-loop --version 2.0.201 --untar
```

3. Set the following variables:

```
Set the environment variables
export HELM_NAME=devops-loop
export NAMESPACE=devops-loop
```

4. Run the following command to create the HCL Container Registry image pull secret:

```
kubectl create secret docker-registry hcl-entitlement-key \
    --docker-username=<accountID>
    --docker-password=<entitlementKey> \
    --docker-server=hclcr.io
    -n ${NAMESPACE}
```

5. Create the Harbor object storage secret by running the following command:

```
kubectl create secret generic harbor-s3-secret \
  -n ${NAMESPACE} \
  --from-literal=accesskey=<ACCESS_KEY> \
  --from-literal=secretkey=<SECRET_KEY>
```

6. Run the following command to export the current Helm values:

```
helm get values ${HELM_NAME} \
  -n ${NAMESPACE} \
  > user_input_values.yaml
```

7. Update the installation script (`hcl-devops-loop/scripts/HCL-loop-XXX-install.sh`) with the following changes:
    - Add placeholders and instructions for the required environment-specific details, including:
        - Domain name
        - License information
        - TLS certificate details
        - Storage class configuration
        - Update both Helm upgrade commands in the installation script to include the user values file:

          ```
          -f user_input_values.yaml
          ```

          This ensures that all user-specific configuration is applied consistently during both Helm upgrade operations.

   For example:

```
helm upgrade --install ${HELM_NAME} \
  oci://hclcr.io/devops-automation-helm/hcl-devops-loop \
  --version ${LOOP_CHART_VERSION} \
  -f user_input_values.yaml \
  --set harbor.enabled=false \
  --hide-notes \
  --timeout 30m \
  -n ${NAMESPACE} || return 1

helm upgrade --install ${HELM_NAME} \
  oci://hclcr.io/devops-automation-helm/hcl-devops-loop \
  --version ${LOOP_CHART_VERSION} \
  -f user_input_values.yaml \
  ${HARBOR_HELM_OPTIONS} \
  --hide-notes \
```

```
  --timeout 30m \
  -n ${NAMESPACE} || return 1
```

8. Update the installation script with the required Harbor configuration:

```
HARBOR_ENABLED=true
HARBOR_S3_BUCKET=<bucket>
HARBOR_S3_REGION=<region>
HARBOR_S3_ENDPOINT=<endpoint>
HARBOR_TRIVY_STORAGE_CLASS=<storage-class>
HARBOR_OIDC_ADMIN_GROUP=<group>
```

9. Run the modified installation script for upgrade.

   For example:

```
bash HCL-loop-K8s-install.sh
```

10. Run the following command to verify that the Harbor pods are running:

```
kubectl get pods -n ${NAMESPACE}
```

11. To confirm that Harbor is accessible and the URL is active, open a web browser and navigate to:

```
https://harbor.<DOMAIN>
```

You have upgraded DevOps Loop with Harbor enabled.

Related reference

Troubleshooting DevOps Loop installation and upgrade

Related information

Troubleshooting Harbor integration

Integrations in DevOps Loop

DevOps Loop provides a set of integrations to extend the platform's capabilities and connect your loops to external systems and intelligent tools. These integrations enable you to automate tasks, enhance productivity, and interact with loop resources from outside the DevOps Loop interface.

AI provider integration

DevOps Loop supports AI-assisted features through external AI providers such as **OpenAI**, **Claude Desktop**, **Gemini**, **IBM watsonx**, and **Ollama**. When configured, these integrations enable Loop Genie to generate summaries, insights, and contextual information across work items, build results, deployments, and other supported capabilities.

MCP server integration

The MCP server (Model Context Protocol Server) allows external MCP-compatible tools such as **VS Code (Copilot)** and **Claude Desktop** to connect directly to DevOps Loop. Through this integration, you can query, retrieve, and modify loop resources from supported external tools. For more information, see DevOps Loop MCP server on page 111.

AI provider integration for Loop Genie - Tech Preview

Disclaimer:

This release contains access to the AI provider configurations for Loop Genie feature in HCL DevOps Loop as a Tech Preview. The Tech Preview is intended for you to view Loop Genie's capabilities by integrating with major LLM providers such as OpenAI, Claude Desktop, Gemini, IBM watsonx, and Ollama and provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

You can configure AI providers such as **OpenAI**, **AZURE OpenAI**, **Claude Desktop**, **Gemini**, **IBM watsonx**, and **Ollama** with DevOps Loop. Configuring any one of these providers activates Loop Genie for your environment. Only one provider can be active at a time.

Loop Genie relies on external AI providers to power natural language understanding, multi-agent orchestration, and conversational task execution. By configuring an AI provider, you enable Loop Genie to process prompts, retrieve and summarize data, and perform supported actions across your DevOps Loop environment.

Loop Genie becomes available once at least one AI provider is configured and active. Switching providers immediately changes which LLM Loop Genie uses for processing prompts. All provider configurations remain saved even when disabled.

With at least one AI provider configured, Loop Genie can:

- Interpret natural language prompts
- Execute multi-step, multi-capability queries
- Retrieve data such as issues, repositories, branches, and commit history
- Perform supported automation tasks (For example: create work items, create branches, list repositories, edit files, manage pull requests)
- Produce structured, readable output
- Provide context-aware responses within a conversational thread.

You can manage integrations that are configured in HCL DevOps Loop by editing, deleting, enabling, or disabling them as required.

You can use the following options on the **Integrations** page:

Actions	Description
Search for an integration	You can search for an integration that is added by entering the name in the **Search plugin** field when the list of integrations is lengthy.
Enable or disable a provider	You can enable or disable an AI provider by clicking the toggle button in the **Provider** column. At any point in time, only one provider can be enabled. Other providers in the list are disabled automatically.
Edit an integration	You can modify the integration details by clicking in the **Actions** column.
View an integration	You can view the integration details by clicking in the **Actions** column.
Delete an integration	You can delete an integration when it is no longer required by clicking in the **Actions** column.

Prerequisites for setting up AI providers for Loop Genie

When you want to configure an AI provider such as OpenAI, IBM watsonx, Gemini, Claude or Ollama with HCL DevOps Loop, ensure that the prerequisites are met.

Requirements for OpenAI integration

Ensure the following steps are completed before integrating OpenAI with DevOps Loop:

1. **Create an OpenAI account**: Sign up on OpenAI's platform and obtain API access.
2. **Generate an API key**: Navigate to the OpenAI dashboard and create a secret API key for authentication.
3. **Install OpenAI library**: Use Python or another programming language to install the OpenAI package.

Requirements for AZURE OpenAI integration

Ensure the following steps are completed before integrating AZURE OpenAI with DevOps Loop:

1. **Create a Model Deployment**
    a. **Sign In:** Log into the Azure Portal.
    b. **Open Resource:** Click on your Azure OpenAI resource from your dashboard.
    c. **Launch Studio:** Select **Go to Azure AI Foundry portal** (or Azure OpenAI Studio).
    d. **Go to Deployments:** Click **Deployments** in the left navigation menu.
    e. **Start Deployment:** Select **+ Deploy model**, then click **Deploy base model**.
    f. **Choose Model:** Select your preferred model (for example, **gpt-4o**) and click **Confirm**.
    g. **Name & Save:** Type a **Deployment name**, click **Deploy**, and save the name for your setup.

Requirements for Claude Desktop integration

Ensure the following steps are completed before integrating Claude Desktop with DevOps Loop:

1. **Create an Claude account**: Sign up on Claude's platform and obtain API access.
2. **Generate an API key**: Navigate to the Claude dashboard and create a secret API key for authentication.
3. **Install Claude library**: Use Python or another programming language to install the Claude Desktop AI package.

Requirements for Gemini integration

Ensure the following steps are completed before integrating Gemini with DevOps Loop:

1. **Create an Gemini account**: Sign up on Gemini's platform and obtain API access.
2. **Generate an API key**: Navigate to the Gemini dashboard and create a secret API key for authentication.
3. **Install Gemini library**: Use Python or another programming language to install the Gemini package.

Requirements for IBM watsonx integration

Ensure that you have the following details:

1. API Key
2. Project ID
3. Endpoint URL

Requirements for Ollama integration

Ensure the following steps are completed before integrating Ollama with DevOps Loop:

1. **Install Ollama**: Download the installer and install it.

   You must have read and understood the system requirements for each model of Ollama. for more information, refer to https://ollama.com.

2. **Download a model**: Download the required large language model (LLM) such as Llama 2 or Mistral.
3. **Set up Python environment**: Create and activate a virtual environment and install dependencies.

   Note:

   If you are using the Ollama instance that is bundled with DevOps Loop, you must follow these steps:

   a. Enable the instance in the Helm chart by setting the value to `true` in the `values.yaml` file:

   ```
   llama:
       enabled: true
   ```

   b. Specify the required large language model (LLM) name in the Helm chart by providing the values in the `values.yaml` file:

   ```
   llama:
       ollama:
   ```

```
models:
pull:
    - <model name>
    - <model name>
run:
    - <model name>
    - <model name>
```

Where, the pull command is used to download a model, and the run command is used to run the model.

Configuring OpenAI integration

You can configure the OpenAI integration in HCL DevOps Loop to connect to the provider and use Loop Genie services.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Completed the steps to prepare your OpenAI account for integration with DevOps Loop. See Requirements for OpenAI integration on page 95.

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click **Add Integration**.
   The **Add Integration** pane is displayed with the **Overview** tab.
3. Enter a name to identify the integration in **Name**.
4. Select **Open AI** from the **Provider** list.
5. Select one of the following options in the **Credential Type** list:
   - API Key: Go to step 6 on page 97.
   - Password: Go to step 7 on page 97.
6. Enter the API key of the AI provider account to authenticate the connection in **API Key**.
7. Enter the details in the following fields:
   - Username
   - Password
   - Confirm Password
8. Click **Next**.
   The **Configure** tab is displayed.
9. Perform the following steps in the Configure tab to set the parameters:
   a. Enter the **API Endpoint** URL used by your OpenAI account.
   b. Select **Chat** in **Mode** to set the mode of communication.
   c. Select an LLM to be used from the **Model** list.
   d. Enter the **Organization ID** configured in your OpenAI account for your organization.
   e. Enter the **Project ID** configured in your OpenAI account for your organization.

f. Enter one or more **Stop Sequences** to define where the model should stop generating text. For example: `"stop"` or `"end"`.
g. Use the **Temperature** slider to control the creativity of the response:
    - Lower values produce more precise, consistent answers.
    - Higher values produce more creative or varied output.
h. Set the **Max Length** slider to define the maximum number of tokens the model can generate.
i. Use the **Top P** slider to adjust the diversity of the output:
    - Lower values focus on the most likely words.
    - Higher values allow more varied responses.
j. Set the **Frequency Penalty** slider to reduce repetition of words or phrases.
    - Higher values result in less repetitive output.
k. Set the **Presence Penalty** slider to encourage the model to introduce new topics rather than repeating previous ones.
l. Set the **Best Of** slider to choose how many internal responses the model generates before returning the best one.

10. Click **Save** to apply the configuration.

Results

You have configured the OpenAI integration with DevOps Loop.

Configuring AZURE OpenAI integration

You can configure the AZURE OpenAI integration in HCL DevOps Loop to connect to the provider and use Loop Genie services.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Created an Azure OpenAI account.
- Created and deployed the required Azure OpenAI model deployment. See Requirements for AZURE OpenAI integration on page 95.
- Ensured that you have the following details:
    - Deployment Name
    - API Endpoint URL
    - API Version

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click the **Add Integration** button.

The **Add Integration** pane is displayed with the **Overview** tab.

3. Enter a name to identify the integration in **Name**.
   For example: AZURE
4. Select **AZURE OpenAI** from the **Provider** list.
5. Select one of the following options in the **Credential Type** list:
   - API Key: Go to step 6 on page 99.
   - Password: Go to step 7 on page 99.
6. Enter the API key of the AI provider account to authenticate the connection in **API Key**.
7. Enter the details in the following fields:
   - Username
   - Password
   - Confirm Password
8. Click **Next**.
   The **Configure** tab is displayed.
9. Perform the following steps in the Configure tab to set the parameters:
   a. Enter the **API Endpoint** URL used by your Azure OpenAI account.

      For example: https://qadevopsplan.openai.azure.com

   b. Select **Chat** in **Mode** to set the mode of communication.
   c. Enter the **Deployment Name**.

      The deployment name is the Azure OpenAI model deployment that you want DevOps Loop to use.

   d. **Optional:** The **Embedding Deployment Name** field is disabled, which is the default setting, and DevOps Loop uses **Keyword (BM25) search**. To enable semantic search, update the frontend ConfigMap to configure the embedding model. After it is enabled, the **Embedding Deployment Name** field is displayed in the integration UI. Enter the deployment name of your Azure OpenAI embedding model in this field.
   a. Enter the **API Version**.
10. Click **Save**.

Results

You have configured the AZURE OpenAI integration with DevOps Loop.

Configuring Claude integration

You can configure the Claude integration in HCL DevOps Loop to enable Loop Genie services by connecting to the Claude Desktop provider.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Completed the steps to prepare your Claude Desktop account for integration with DevOps Loop. See Prerequisites for setting up AI providers for Loop Genie on page 95.

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click **Add Integration**.
   The **Add Integration** pane is displayed with the **Overview** tab.
3. Enter a name to identify the integration in **Name**.
4. Select **Claude** from the **Provider** list.
5. Select one of the following options in the **Credential Type** list:
   - API Key: Go to step 6 on page 100.
   - Password: Go to step 7 on page 100.
6. Enter the API key of your Claude account to authenticate the connection in **API Key** and go to step 8 on page 100.
7. Enter the details in the following fields:
   - Username
   - Password
   - Confirm Password
8. Click **Next**.
   The **Configure** tab is displayed.
9. Perform the following steps in the **Configure** tab to set the parameters:
   a. Enter the **API Endpoint** URL used by your Claude account.
      - URL = https://api.anthropic.com
   b. Select **Chat** in **Mode** to set the mode of communication.
   c. Select an LLM to be used from the **Model** list.
   d. Enter the **Organization ID** configured in your Claude account for your organization.
   e. Enter the **Project ID** configured in your Claude account for your organization.
   f. Enter one or more **Stop Sequences** to define where the model should stop generating text.

      For example: `"stop"` or `"end"`.

   g. Use the **Temperature** slider to control the creativity of the response:

      - Lower values produce more precise, consistent answers.
      - Higher values produce more creative or varied output.

   h. Set the **Max Length** slider to define the maximum number of tokens the model can generate.
   i. Use the **Top P** slider to control the diversity of the generated output:

      - Lower values focus on the most likely words.
      - Higher values allow more varied responses.

   j. Set the **Frequency Penalty** slider to reduce repetition of words or phrases.
      - Higher values result in less repetitive output.

k. Set the **Presence Penalty** slider to encourage the model to introduce new topics rather than repeating previous ones.
l. Set the **Best Of** slider to choose how many internal responses the model generates before returning the best one.

10. Click **Save** to apply the configuration.

Results

You have configured the Claude with DevOps Loop.

Configuring Gemini integration

You can configure Gemini integration in HCL DevOps Loop to connect to the provider and use Loop Genie services.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Completed the steps to prepare your Gemini account for integration with DevOps Loop. See Requirements for Gemini integration on page 96.

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click **Add Integration**.
   The **Add Integration** pane is displayed with the **Overview** tab.
3. Enter a name to identify the integration in **Name**.
4. Select **Gemini** from the **Provider** list.
5. Select one of the following options in the **Credential Type** list:
   - API Key: Go to step 6 on page 101.
   - Password: Go to step 7 on page 101.
6. Enter the API key of your Gemini account to authenticate the connection in **API Key** and go to step 8 on page 101.
   - Name = `Google_API_Key`
   - Value = `AIzaSyDySjxBpUlHqUhdIvh6VnXbxOYBJ5wout4`
7. Enter the details in the following fields:
   - Username
   - Password
   - Confirm Password
8. Click **Next**.
   The **Configure** tab is displayed.
9. Perform the following steps in the **Configure** tab to set the parameters:
   a. Enter the **API Endpoint** URL used by your Gemini account.
      - URL = https://generativelanguage.googleapis.com
   b. Select **Chat** in **Mode** to set the mode of communication.
   c. Select an LLM to be used from the **Model** list.

d. Enter the **Organization ID** configured in your Gemini account for your organization.
e. Enter the **Project ID** configured in your Gemini account for your organization.
f. Enter one or more **Stop Sequences** to define where the model should stop generating text. For example: `"stop"` or `"end"`.
g. Use the **Temperature** slider to control the creativity of the response:
    - Lower values produce more precise, consistent answers.
    - Higher values produce more creative or varied output.
h. Set the **Max Length** slider to define the maximum number of tokens the model can generate.
i. Use the **Top P** slider to adjust the diversity of the output:
    - Lower values focus on the most likely words.
    - Higher values allow more varied responses.
j. Set the **Frequency Penalty** slider to reduce repetition of words or phrases.
    - Higher values result in less repetitive output.
k. Set the **Presence Penalty** slider to encourage the model to introduce new topics rather than repeating previous ones.
l. Set the **Best Of** slider to choose how many internal responses the model generates before returning the best one.

10. Click **Save** to apply the configuration.

Results

You have configured the Gemini integration with DevOps Loop.

Configuring IBM watsonx integration

You can configure the IBM watsonx integration in HCL DevOps Loop to connect to the provider and use Loop Genie services.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Ensured that you have details such as API Key, Project ID, and Endpoint URL.

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click **Add Integration**.
   The **Add Integration** pane is displayed with the **Overview** tab.
3. Perform the following steps in the **Overview** tab:
    a. Enter a name to identify the integration in the **Name** field.
    b. Select **Watson X** in the **Provider** list.

c. Select the API key in the **Credentials** list.

d. Enter the API Key in the **API Key** field.

e. Click **Next**.

The **Configure** tab is displayed.

4. Perform the following steps in the **Configure** tab:

a. Enter the API Endpoint URL in the **API Endpoint** field.

b. Enter the release version of the IBM watsonx.ai model in the **Version** field to specify the model to which DevOps Loop should connect.

c. Select a **Model** from the list.

d. Enter the project ID in the **Project ID** field.

e. Enter the string to be used to stop the generation of text by AI in the **Stop Sequences** field.

For example, you can set "stop" or "end" as the stop sequence.

f. Select a **Decoding** method from the list to control how the model generates output.
The following options are available:
- **Greedy:** Produces the most likely next token at each step for straightforward output.
- **Sample:** Randomly selects from probable next tokens, resulting in more varied and creative responses.

g. Use the **Repetition Penalty** slider to control how much the model discourages repeating the same tokens in its output:
For example: A higher value, for example, 1.5, makes more different responses. A lower value, for example, 1.0, allows natural repetition of words.

h. Specify the minimum number of tokens in the **Min Token** for the response.

i. Specify the maximum number of tokens in the **Max Token** for the response.

j. Click **Save**.

Results

You have configured the IBM watsonx integration with DevOps Loop.

Configuring Ollama integration

You can configure the Ollama integration in HCL DevOps Loop to connect to the provider and use Loop Genie services.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Completed the steps to prepare your Ollama account for integration with DevOps Loop. See Requirements for Ollama integration on page 96.

1. Click **Settings > Installed Integrations**.
   The **Installed Integrations** page is displayed.
2. Click **Add Integration**.
   The **Add Integration** pane is displayed with the **Overview** tab.
3. Enter a name to identify the integration in **Name**.
4. Select **Ollama** from the **Provider** list.
5. Click **Next**.
   The **Configure** tab is displayed.
6. Enter the URL of the AI provider in the **API Endpoint** field.
7. Select **Chat** in **Mode** to set the mode of communication.
8. Select an LLM to be used from the **Model** list.
9. Click **Save**.

Results

You have configured Ollama integration with DevOps Loop.

Harbor integration overview

Harbor is an integrated capability in DevOps Loop that provides an OCI-compliant container registry, vulnerability scanning, image signing, and image management capabilities.

DevOps Loop and Harbor are installed together in one installation process. You must set the `HARBOR_ENABLED=true` parameter when installing DevOps Loop.

Harbor capabilities

Harbor provides the following capabilities when it is integrated with DevOps Loop:

- Container image storage and distribution
- Role-based access control
- Vulnerability scanning through Trivy
- OpenID Connect (OIDC) authentication through Keycloak
- Image replication and lifecycle management
- Image bootstrapping support

Architecture

Harbor uses the following services and infrastructure components when it is integrated with DevOps Loop:

- Keycloak for authentication and supplies group claims
- PostgreSQL for metadata storage
- Valkey for caching
- S3-compatible object storage for image artifacts
- Kubernetes persistent storage for Harbor services

Authentication

Harbor uses the same Keycloak realm and single sign-on session as DevOps Loop. You can authenticate through the configured identity provider and can access Harbor without maintaining separate credentials.

Prerequisites for Harbor integration

You must ensure that the required Kubernetes platform components, storage configuration, object storage, DNS, TLS, and dependency secrets are available before you install DevOps Loop with Harbor integration.

Kubernetes platform components

The Kubernetes cluster must be accessible and configured with the required tooling for deployment. You must ensure that your Kubernetes environment includes the required platform components.

Before deploying DevOps Loop, ensure that your Kubernetes environment meets the following prerequisites:

- Helm and kubectl access are configured.
- Emissary ingress controller is installed and available.
- Image pull secrets are configured for the required container images.
- A dedicated namespace for DevOps Loop deployment is available.
- If you are deploying DevOps Loop on **OpenShift**, ensure that a cluster administrator is available to configure the required Harbor **Security Context Constraints (SCC)** before Harbor is enabled.

These components support application deployment, ingress routing, and container image access during installation.

DNS configuration for Harbor

You must create a DNS entry for the Harbor endpoint in your DNS provider.

Example values:

- Name: `harbor.<DOMAIN>`
- Type: A or CNAME
- Value: <IP_ADDRESS or LOAD_BALANCER_ENDPOINT>

Ensure that the DNS record is resolvable from the cluster environment and external clients. This configuration is required for Harbor access and OIDC authentication flows.

TLS configuration for Harbor

You must ensure that a valid TLS certificate is available for secure access to Harbor.

The certificate must include the Harbor hostname, for example, `harbor.<DOMAIN>`.

You can use a wildcard certificate, such as `*.example.com` or a SAN certificate that covers both DevOps Loop and Harbor endpoints.

TLS configuration must be compatible with the existing DevOps Loop Ingress setup.

Storage configuration for Harbor

You must ensure that the following storage classes are available in the Kubernetes cluster:

- A `ReadWriteOnce` (RWO) storage class for standard persistence.
- A `ReadWriteMany` (RWX) storage class for Trivy vulnerability scanning.

You must verify the available storage classes by running the following command:

```
kubectl get storageclass
```

The RWX storage class must support shared access across multiple pods and be compatible with Trivy scanning requirements.

Object storage configuration for Harbor

You must ensure that an S3-compatible object storage system is available. Supported storage providers are as follows:

- AWS S3
- IBM Cloud Object Storage
- Google Cloud Storage (HMAC-enabled)
- MinIO
- Other S3-compatible storage systems

You must ensure that the following configuration values are available:

- Bucket name
- Region
- Endpoint URL
- Access key
- Secret key

These values are required for configuring Harbor registry storage.

Note: If you use Google Cloud Storage as the object storage backend, you must configure HMAC keys because Harbor requires S3-compatible authentication. You must also run the `gcloud storage buckets update ... --no-uniform-bucket-level-access` command to disable Uniform Bucket-Level Access.

Harbor dependency secrets

The following secrets are required during Harbor installation:

- `devops-loop-valkey`
- `devops-loop-postgresql`

These secrets are automatically created during the DevOps Loop installation.

The Harbor installation validates the availability of these secrets before it proceeds.

You have prepared all the required prerequisites for installing DevOps Loop with Harbor integration.

You can now proceed with installing DevOps Loop with Harbor integrated. See Installing DevOps Loop with Harbor integration on page 107.

Installing DevOps Loop with Harbor integration

You can configure Harbor as an integrated capability during DevOps Loop installation to provide container image storage, vulnerability scanning, and image management capabilities.

Before you begin

You must have completed the prerequisites tasks. See Prerequisites for Harbor integration on page 105.

1. Create the object storage secret.

   ```
   kubectl create secret generic harbor-s3-secret \
     -n devops-loop \
     --from-literal=accesskey='<ACCESS_KEY>' \
     --from-literal=secretkey='<SECRET_KEY>'
   ```

   Verify that the secret is created:

   ```
   kubectl get secret harbor-s3-secret -n devops-loop
   ```

2. **Optional:** Configure MinIO if you use MinIO as the object storage provider.
   - Install MinIO.
   - Create a bucket, for example, `loopharbor`.
   - Ensure that the bucket is accessible.

   If you deploy DevOps Loop in an air-gapped environment, ensure that the required container images are available in your internal registry before you install Harbor. After the installation is complete, you can bootstrap commonly used images into Harbor by using the bootstrap script.

3. Set the following parameter to configure Harbor with DevOps Loop:

Enable or disable Harbor:

```
HARBOR_ENABLED=true
HARBOR_S3_BUCKET=<BUCKET_NAME>
HARBOR_S3_REGION=<REGION>
HARBOR_S3_ENDPOINT=<S3_ENDPOINT>
HARBOR_TRIVY_STORAGE_CLASS=<RWX_STORAGE_CLASS>
```

Optional parameter:

```
HARBOR_OIDC_ADMIN_GROUP=<group-name>
```

Default admin group: `harbor-admins`.

Note: You can customise the Harbor admin group name in the optional parameter in case, the group name is not provided during the installation, the group will be created with the default name harbor-admins. Users who require Harbor system administrator access must be members of that Keycloak group.

4. **Optional**: Perform the following steps to configure the Harbor Security Context Constraints (SCC) if you are installing DevOps Loop on OpenShift:
   a. Apply the Harbor SCC that is provided with the installation.

   ```
   scripts/harbor/openshift/harbor-scc.yaml
   ```

   b. Run the helper script to apply the SCC and bind it to the Harbor service account:

   ```
   scripts/harbor/openshift/apply-harbor-scc.sh \
     --namespace devops-loop \
     --service-account harbor \
     --scc-name harbor-uid-10000-devops-loop
   ```

   The Harbor SCC is not installed automatically by Helm because it is stored under `scripts/harbor/openshift/` rather than the Helm `templates/ directory`. Before Harbor is enabled, the installer verifies that the SCC exists and is correctly bound to the Harbor service account.

   Note: If `https://harbor.<DOMAIN>` is not resolvable in your OpenShift environment, set `HARBOR_EXTERNAL_URL` to a DNS-resolvable OpenShift router wildcard URL.

5. Run the installation script in the Helm README for installing DevOps Loop.
   For example: `bash <Install-script>.sh`

   The installer performs the following tasks:

   - Deploys DevOps Loop.
   - Validates prerequisites.
   - Validates secrets.
   - Installs Harbor when enabled.
   - Configures OIDC.
   - Configures ingress and HTTPS.

Note: When `HARBOR_ENABLED=true`, the installer uses a two-phase Helm installation process. The Helm release history can show an initial revision with an Install status and a later revision with an Upgrade status. This behavior is expected.

6. Run the following command to verify that the Harbor pods are running:

   ```
   kubectl get pods -n devops-loop | grep harbor
   ```

   The output must include the following Harbor pods:

   - `harbor-core`
   - `harbor-jobservice`
   - `harbor-nginx`
   - `harbor-portal`
   - `harbor-registry`
   - `harbor-trivy`

Results

You have installed DevOps Loop with Harbor enabled.

What to do next

You can perform the following tasks:

- Access Harbor from the DevOps Loop. See Accessing Harbor from DevOps Loop on page 109.
- Import images into Harbor. See Importing and validating images in Harbor registry on page 110.

Accessing Harbor from DevOps Loop

You can access Harbor from the DevOps Loop when it is configured with DevOps Loop installation.

Before you begin

You must have completed the following tasks:

- Installed Harbor and ensured it is running in the `devops-loop` namespace.
- Esnured that you are signed in to DevOps Loop.

1. Select the application launcher icon .
   The application launcher displays the DevOps Loop capabilities.
2. Select **Harbor**.
   The Harbor login page opens in a new tab.
3. Select **LOGIN VIA OIDC PROVIDER**.
4. Authenticate by using your Keycloak credentials.

Harbor uses the same Keycloak realm and single sign-on session as DevOps Loop.

Note: Logging out of Harbor clears the Harbor application session. It does not log the user out of DevOps Loop. If the DevOps Loop or Keycloak SSO session is still active, clicking Login with Keycloak in Harbor again may automatically log in the same user without asking for credentials.

To force a credential prompt, the user must also log out fromDevOps Loop or Keycloak, clear browser cookies, use an incognito/private browser session, or use a separate browser profile

Results

You can access Harbor from the application launcher and sign in through OIDC.

Related information

Managing Users

Assign users to a project

Importing and validating images in Harbor registry

You can import images into Harbor to preload repositories with commonly used container images.

Before you begin

You must have completed the following tasks:

- Installed Harbor and ensured it is accessible. See Accessing Harbor from DevOps Loop on page 109.
- Installed skopeo on the system where you run the bootstrap script.

1. Update the bootstrap image list.

   Edit the `scripts/harbor/bootstrap-images.txt` file in the following format:

   ```
   source_image target_project target_repository target_tag
   ```

   For example:

   ```
   docker://docker.io/library/nginx:latest devops nginx bootstrap
   docker://docker.io/library/alpine:latest devops alpine bootstrap
    docker://docker.io/library/busybox:latest devops busybox bootstrap
   ```

2. Run the bootstrap script.

   ```
   DOMAIN=<DOMAIN> \
   NAMESPACE=devops-loop \
   RELEASE=devops-loop \
   scripts/harbor/bootstrap-images.sh
   ```

3. Verify that the imported images are available in Harbor.
4. Validate image push and pull operations:

```
docker login harbor.<DOMAIN>
docker tag nginx:latest harbor.<DOMAIN>/devops/nginx:test
docker push harbor.<DOMAIN>/devops/nginx:test
docker pull harbor.<DOMAIN>/devops/nginx:test
```

5. Perform the following steps to verify vulnerability scanning:
    a. Open Harbor.
    b. Navigate to the project that contains the uploaded image.
    c. Enable **Automatically scan images on push** for the project if it is not already enabled.
    d. Push a container image to the project.
    e. Verify that Harbor displays the vulnerability scan results for the image.

   Harbor scans the image by using Trivy and displays the detected vulnerabilities.

Results

You have imported the specified images into Harbor and they are available for use.

DevOps Loop MCP server

The DevOps Loop Model Context Protocol (MCP) server provides a standardized interface so that external MCP clients can interact with DevOps Loop and perform supported operations.

The MCP-compatible clients, such as Visual Studio Code (through MCP extensions) and Claude Desktop, can use this interface to query, retrieve, and modify loop resources directly from their development or AI-assisted environments.

By using the MCP server, you can access DevOps Loop capabilities without switching between interfaces. It enforces secure and controlled access by honoring the permissions defined in DevOps Loop.

The server maintains session security through a 30-minute idle timeout. If no activity occurs during this period, the session expires and requires reauthentication. After initial configuration, MCP client settings are stored persistently. MCP clients then reconnect automatically in subsequent sessions.

The MCP server offers a set of domain-specific tools that allow external clients to perform the following operations:

- Managing loops
- Accessing tenant and teamspace information
- Querying analytics data
- Performing deployment and build operations
- Managing repositories and pull requests
- Interacting with planning and test resources

Each tool defines its required inputs and expected outputs, ensuring consistent and predictable interactions.

Related information

Getting started with the DevOps Loop MCP server on page 112

Getting started with the DevOps Loop MCP server

Use the Loop MCP endpoint to fetch and manage loop platform entities, including loops, teamspaces, users, and loop analytics.

Each DevOps Loop capability, which includes Loop platform, Plan, Measure, Control, Deploy, Build, and Test exposes a dedicated MCP endpoint. You must configure your MCP client with the appropriate endpoint URL based on the capability that you want to access.

After you configure the endpoint, the MCP client establishes a session with the DevOps Loop MCP server and makes the available tools accessible within the client interface.

DevOps Loop MCP endpoint

You can use the following endpoint to connect to the DevOps Loop MCP server:

```
https://<loop_hostname>/loop/mcp
```

MCP tools available in DevOps Loop

After an MCP client is connected to DevOps Loop, it can access a set of tools offered by DevOps Loop. These tools provide controlled access to loop resources across the loop server, tenant server, and analytics services. Each tool specifies its required inputs and expected output.

Use these tools when you need loop and teamspace administration, user membership, or cross-domain analytics queries. Do not use these tools for Plan work item lifecycle updates, repository pull request actions, build execution control, deployment execution, or test execution.

Parameter conventions:

- `teamspace_id` and `loop_id` must be valid UUID values.
- `user` must be a valid email address.
- For analytics date filters, use `YYYY-MM-DD`.
- For filtered searches, pass only supported filter keys for the target entity.

Table 1. Loop-related tools

MCP Tools	Description	Input	Output
loop_get_all_loops	Fetch all loops in a teamspace.	teamspace_id	List of loops that you are a member of.
loop_get_all_loop_users	Fetch all users in a loop.	loop_id, teamspace_id	List of users who are part of the loop.

Table 1. Loop-related tools (continued)

MCP Tools	Description	Input	Output
loop_add_user_to_loop	Add a user to a loop.	loop_id, teamspace_id, user (email)	Confirmation that the user was added to the loop.
loop_add_test_project_to_loop	Add a test project to a loop.	teamspace_id, loop_id, project_name, project_description	Details of the test project created in the loop.
loop_add_control_repo_to_loop	Add a control repository to a loop.	teamspace_id, loop_id, repo_name	Details of the control repository added to the loop.

Table 2. Loop tenant-related tools

Tools	Description	Input	Output
loop_get_all_tenants	Fetch all tenants.	None	List of all tenants
loop_get_all_teamspace_users	Fetch all users in a teamspace.	teamspace_id	List of users in the teamspace
loop_get_all_teamspaces	Fetch all teamspaces.	None	List of all teamspaces
loop_invite_user_to_teamspace	Invite a user to a teamspace.	teamspace_id, user (email)	Confirmation that the user was invited to the teamspace

Table 3. Loop analytics-related tools

Command	Description	Input	Output
analytics_search_work_items	Searches work items by using filters.	teamspace_id, loop_id, optional filters	List of work items matching the filters.
analytics_get_work_item_status	Retrieves status for work items.	teamspace_id, loop_id, filters	List of work item statuses.
analytics_count_work_items	Counts work items by using filters.	teamspace_id, loop_id, filters	Count of work items matching the filters.
analytics_get_story_points	Retrieves story point analytics.	teamspace_id, loop_id, owner, date filters	Story point analytics data.
analytics_get_recent_and_old_work_items	Retrieves recent and oldest work items.	teamspace_id, loop_id, optional filters	List of work items ordered by time.

Table 3. Loop analytics-related tools (continued)

Command	Description	Input	Output
analytics_search_commits	Search commits by creator, date, repo, etc.	teamspace_id, loop_id, filters	List of commits matching the filters.
analytics_search_pull_requests	Attribute-based PR search.	teamspace_id, loop_id, filters	List of pull requests matching the filters.
analytics_count_pull_requests	Count PRs by creator, repo, status, or month.	teamspace_id, loop_id, filters	Count of pull requests matching the filters.
analytics_get_recent_and_old_pull_requests	Retrieves time-ordered pull requests.	teamspace_id, loop_id, filters	List of pull requests ordered by time.
analytics_search_builds	Searches builds by using filters.	teamspace_id, loop_id, filters	List of builds matching the filters.
analytics_count_builds	Counts builds by creator, status, or month.	teamspace_id, loop_id, filters	Count of builds matching the filters.
analytics_get_recent_and_old_builds	Retrieves recent and oldest builds.	teamspace_id, loop_id, filters	List of builds ordered by time.
analytics_search_deployments	Search deployments with filters.	teamspace_id, loop_id, filters	List of deployments matching the filters.
analytics_count_deployments	Counts deployments by status, creator, or month.	teamspace_id, loop_id, filters	Count of deployments matching the filters.
analytics_get_recent_and_old_deployments	Retrieves recent and oldest deployments.	teamspace_id, loop_id, filters	List of deployments ordered by time.
analytics_field_values	Retrieves available values for discoverable fields of an analytics entity before applying filters.	entity, teamspace_id, loop_id, optional fields	Lists of available field values, unsupported fields, and fields containing blank values.

Plan MCP endpoint and tools

Use Plan MCP tools to create, read, update, and organize Agile planning data, including work items, projects, releases, sprints, approvals, and metadata.

Plan MCP endpoint

You can use the following endpoint to connect to the Plan MCP server:

```
https://<loop_hostname>/plan/mcp
```

MCP tools available in Plan

Use these tools when you need planning operations, such as managing work items, assigning ownership, linking hierarchy, setting sprint or release values, or listing projects, releases, and sprints. Do not use these tools for repository or pull request actions, build execution, deployment execution, test execution, or value stream analytics reporting.

Parameter conventions:

- `applicationName` must match an accessible Plan application exactly.
- `teamspaceId` is required for scoped operations and must be a valid UUID.
- Resolve `teamspaceId` by using `plan_getDevopsPlanApplications` when missing.
- Use `YYYY-MM-DD` for date values such as startDate and endDate.
- Use positive integers for `pageNumber` and `pageSize`.

Tools	Description	Input	Output
plan_deleteWorkitem	Deletes an existing work item from the specified Plan application.	applicationName, workItemId, teamspaceId	Confirmation that the work item was successfully deleted.
plan_fetchWorkitem	Retrieves detailed information for a specific work item by its ID.	workItemId, teamspaceId, applicationName	A record containing all available details for the specified work item.
plan_getAgileApprovals	Retrieves a list of existing Agile approval records from a Plan application.	applicationName, teamspaceId	A list of approval records.
plan_getAgileReleases	Retrieves a list of existing Agile Releases from a Plan application.	applicationName, teamspaceId	A list of Agile releases.
plan_getAgileSprints	Retrieves a list of existing Agile Sprints from a Plan application.	applicationName, teamspaceId	A list of Agile sprints.

plan_getAgileWorkitems	Retrieves Agile work items, supporting sprint and release filtering.	applicationName, teamspaceId, optional filters	A list of Agile work items matching the filters.
plan_getAvailableAgileWorkItemTypes	Retrieves a list of existing Agile work item types for a specific project.	applicationName, teamspaceId, project	A list of Agile work item types.
plan_getDevopsPlanApplications	Retrieves all DevOps Plan applications available to the current user.	None	A list of accessible Plan applications, including name and teamspaceId.
plan_getProjectsInAgileApplication	Retrieves all projects from a specified Plan application.	applicationName, teamspaceId	Project details, including ID, name, and description.
plan_getWorkItemMetaData	Retrieves read-only Work Item Metadata (schema and field definitions).	applicationName, teamspaceId	Work Item Metadata schema (field names, types, allowed values, etc.).
plan_getWorkItemStatesAndStateTransitions	Retrieves all available work item state transitions in the specified Plan application.	applicationName, teamspaceId	Detailed transition information (Action name, Destination state, Source states).
plan_addChildToParentWorkitem	Adds/links a child work item to a parent work item.	applicationName, parentWorkItemId, childWorkItemId, teamspaceId	Confirmation of successful linking.
plan_addCommentToWorkitem	Adds a comment to a work item.	applicationName, workItemId, comment, teamspaceId	Confirmation that the comment was added.
plan_addSubscriberToWorkitem	Adds a subscriber to a work item.	applicationName, workItemId, username, teamspaceId	Confirmation that the user was added as a subscriber.
plan_assignOwnerToWorkitem	Assigns or reassigns an owner to a work item.	applicationName, workItemId, owner, teamspaceId	Confirmation that the owner was successfully assigned.

plan_changeResolutionOfWorkitem	Changes the Resolution field of a work item.	applicationName, workItemId, resolution, teamspaceId	Confirmation that the resolution was changed.
plan_changeStateOfWorkitem	Changes or sets the State field of a work item.	applicationName, workItemId, state, teamspaceId	Confirmation that the state was changed.
plan_createAgileWorkitem	Creates a new Agile work item.	applicationName, title, type, project, teamspaceId	Details of the newly created work item.
plan_modifyAgileWorkitem	Modifies an existing Agile work item.	applicationName, workItemId, teamspaceId	Confirmation that the work item was successfully modified.
plan_removeSubscriberFromWorkitem	Removes a subscriber from a work item.	applicationName, workItemId, username, teamspaceId	Confirmation that the user was removed as a subscriber.
plan_setPlannedReleaseOfWorkitem	Sets the PlannedRelease field of a work item.	applicationName, workItemId, plannedRelease, teamspaceId	Confirmation that the planned release was set.
plan_setSprintForWorkitem	Sets the Sprint field of a work item.	applicationName, workItemId, sprint, teamspaceId	Confirmation that the sprint was set.
plan_setStoryPointsForWorkitem	Sets the StoryPoints field of a work item.	applicationName, workItemId, storyPoints, teamspaceId	Confirmation that the story points were set.
plan_setWorkitemStartAndEndDates	Sets both StartDate and EndDate fields of a work item.	applicationName, workItemId, startDate, endDate, teamspaceId	Confirmation that the start and end dates were set.
plan_addComponentToProject	Adds or links a component to an existing Agile project.	applicationName, teamspaceId, project, component	Confirmation that the component was added to the project.

plan_addReleaseToProject	Adds, links, or associates an existing release to an existing Agile project.	applicationName, teamspaceId, project, releaseName	Confirmation that the release was associated with the project.
plan_addSprintToProject	Adds, links, or associates an existing sprint to an existing Agile project.	applicationName, teamspaceId, project, sprintName	Confirmation that the sprint was associated with the project.
plan_addSprintToRelease	Adds, links, or associates an existing sprint to an existing Agile Release.	applicationName, teamspaceId, releaseIdentifier, sprintName	Confirmation that the sprint was associated with the release.
plan_addWorkItemTypeToProject	Adds or links a work item type to an existing Agile project.	applicationName, teamspaceId, project, workItemType	Confirmation that the work item type was added to the project.
plan_createAgileProject	Creates a new Agile project in the specified Plan application.	applicationName, teamspaceId, projectName	Details of the newly created project.
plan_createAgileRelease	Creates a new Agile Release in the specified Plan application.	applicationName, teamspaceId, releaseName	Details of the newly created release.
plan_createAgileSprint	Creates a new Agile Sprint in the specified Plan application.	applicationName, teamspaceId, sprintName	Details of the newly created sprint.
plan_modifyAgileRelease	Renames and/or updates an existing Agile Release.	applicationName, teamspaceId, releaseIdentifier	Confirmation that the release was successfully modified.
plan_modifyAgileSprint	Renames an existing Agile Sprint and/or updates its start and end dates within the specified Plan application.	applicationName, teamspaceId, sprintIdentifier	Confirmation that the sprint was successfully modified.
plan_removeComponentFromProject	Removes or unlinks or dissociates an existing Component from an existing	applicationName, teamspaceId, project, component	Confirmation that the component was

	Agile project within the specified Plan application.		removed from the project.
plan_removeReleaseFromProject	Removes or unlinks or dissociates an existing Release from an existing Agile project within the specified Plan application.	applicationName, teamspaceId, project, releaseName	Confirmation that the release was removed from the project.
plan_removeSprintFromProject	Removes or unlinks or dissociates an existing Sprint from an existing Agile project within the specified Plan application.	applicationName, teamspaceId, project, sprintName	Confirmation that the sprint was removed from the project.
plan_removeSprintFromRelease	Removes or unlinks an existing Sprint from an existing Agile Release within the specified Plan application.	applicationName, teamspaceId, releaseIdentifier, sprintName	Confirmation that the sprint was removed from the release.
plan_removeWorkItemTypeFromProject	Removes or unlinks or dissociates an existing work item type from an existing Agile project within the specified Plan application.	applicationName, teamspaceId, project, workItemType	Confirmation that the work item type was removed from the project.

Control MCP endpoint and tools

Use Control MCP tools to manage source control resources and collaboration workflows, including repositories, branches, files, commits, and pull requests.

MCP endpoints

You can use the following endpoint to connect to the Control MCP server:

```
https://<loop_hostname>/control/mcp
```

MCP tools available in Control

Use these tools when you need repository content updates, branch management, pull request operations, or repository search and discovery. Do not use these tools for planning entities, build execution, deployment execution, test execution, or value stream analytics reporting.

Parameter conventions:

- `owner` and `repo` are required for repository-scoped operations.
- `index` is the pull request number and must be a positive integer.
- `filePath` must be a repository-relative path.
- `sha` is required for safe file updates.
- Use positive integers for pagination fields such as `page`, `pageSize`, or `page_size`.

Tools	Description	Input	Output
control_approve_pull_request	Approves a pull request.	owner, repo, index	Confirmation that the pull request was approved.
control_assign_pull_request	Assigns a pull request to a user.	owner, repo, index, assignee(control username)	Confirmation that the pull request was assigned to the user.
control_create_branch	Creates a branch.	owner, repo, branch, old_branch	Confirmation that the branch was successfully created.
control_create_file	Creates a file.	owner, repo, filePath, content, message, branch_name	Confirmation that the file was successfully created and committed.
control_create_pull_request	Creates a pull request.	owner, repo, title, body, head, base	Details of the created pull request.
control_create_repo	Creates a repository in personal account or organization.	name	Details of the created repository.
control_delete_branch	Deletes a branch.	owner, repo, branch	Confirmation that the branch was successfully deleted.
control_delete_file	Deletes a file.	owner, repo, filePath, message, branch_name	Confirmation that the file was successfully deleted.
control_fork_repo	Creates a fork of a repository.	user, repo	Details of the forked repository.
control_get_dir_content	Retrieves a list of entries in a directory.	owner, repo, ref, filePath	List of directory entries (files and directories).

control_get_file_content	Retrieves file content and metadata.	owner, repo, ref, filePath	File content and metadata.
control_get_mcp_server_version	Retrieves the Control MCP server version.	None	Control MCP Server Version.
control_get_pull_request_by_index	Retrieves a pull request by index.	owner, repo, index	Details of the specified pull request.
control_list_branches	Lists branches.	owner, repo	List of branches in the repository.
control_list_my_repos	Lists repositories for the authenticated user.	page, pageSize	List of repositories.
control_list_pull_request_comments	Lists all comments from a pull request (includes general comments and code review comments).	owner, repo, index	List of pull request comments.
control_list_repo_commits	Lists repository commits.	owner, repo, page, page_size	List of repository commits.
control_list_repo_pull_requests	Lists repository pull requests.	owner, repo	List of repository pull requests.
control_merge_pull_request	Merges pull request.	owner, repo, index, merge_title	Confirmation that the pull request was successfully merged.
control_search_org_teams	Searches organization teams.	org, query	List of organization teams.
control_search_repos	Searches repos.	keyword	List of repositories matching the search criteria.
control_search_users	Searches users.	keyword	List of users matching the search criteria.
control_update_file	Updates a file.	owner, repo, filePath, sha, content, message, branch_name	Confirmation that the file was successfully updated and committed.

control_get_my_user_info	Retrieves authenticated user information.	None	User's information.
control_get_user_orgs	Retrieves organizations associated with the authenticated user.	None	List of organizations.
control_get_user_pending_approvals	Lists pending approvals for the authenticated user.	None	List of pending approvals.

Deploy MCP endpoint and tools

Use Deploy MCP tools to execute and monitor application deployments by listing applications, snapshots, environments, and deployment processes.

Deploy MCP endpoint

You can use the following endpoint to connect to the Deploy MCP server:

```
https://<loop_hostname>/deploy/mcp
```

MCP tools available in Deploy

Use these tools when you need to deploy an application snapshot to an environment and monitor deployment status. Do not use these tools for build orchestration, repository pull request actions, planning updates, or test execution.

Parameter conventions:

- `applicationId` scopes application-level listing and deployment operations.
- `applicationProcessId`, `snapshotId`, and `environmentId` are required to start a deployment.
- `requestId` is required to query deployment status.
- Use IDs returned by list operations and treat them as opaque values.

Tools	Description	Input	Output
deploy_snapshot_to_environment	Deploys a snapshot to a target environment.	applicationId, applicationProcessId, snapshotId, environmentId	Confirmation of successful deployment and deployment details.
get_deployment_status	Retrieves the status of a previously requested application deployment.	requestId	The status and results of the requested application deployment.
list_application_environments	Lists the environments for a specific application.	applicationId	List of application environments with names and IDs.

list_application_processes	Lists the deployment processes for a specific application	applicationId	List of deployment processes with names and IDs for the application.
list_application_snapshots	Lists the snapshots for a specific application.	applicationId	List of snapshots for a specific application.
list_applications	Lists applications.	None	List of all available applications.

Build MCP endpoint and tools

Use Build MCP tools to trigger and inspect CI build executions, including projects, workflows, build runs, logs, and retries.

Build MCP endpoint

You can use the following endpoint to connect to the Build MCP server:

```
https://<loop_hostname>/build/mcp
```

MCP tools available in Build

Use these tools when you need to run a build, inspect build details, retrieve build logs, or restart a failed build run. Do not use these tools for deployment execution, test execution, repository pull request actions, or planning data management.

Parameter conventions:

- `project_id` is required for project-scoped build operations.
- `process_id` is required to trigger a process build.
- `build_life_id` is required to fetch run details, logs, or restart a build run.
- Use only IDs returned by Build list endpoints; treat identifiers as opaque values.

Tools	Description	Input	Output
build_list_projects	Lists all projects in DevOps Build.	None	List of projects in DevOps Build.
build_list_workflows	Lists all workflows for a specific project in DevOps Build.	project_id	List of workflows for the specified project.
build_trigger_process_build	Triggers a new build for a specific process in a project using the process trigger API.	project_id, process_id	Request ID for tracking the triggered build.

build_list_build_lives	Lists all build lives for a specific project in DevOps Build.	project_id	List of all build lives for the specified project.
build_get_build_life_info	Retrieves detailed information about a build life in DevOps Build.	build_life_id	Detailed information about the specified build life.
build_get_build_life_logs	Retrieves logs for a specific build life in DevOps Build.	build_life_id	Logs for the specified build life.
build_restart_build_life	Restarts a build run by using the build life ID in DevOps Build.	build_life_id	Confirmation that the build life was successfully restarted.

Test MCP endpoint and tools

Use Test MCP tools to execute tests and retrieve test results in DevOps Test Hub, including project discovery, test listing, test run initiation, and result lookup.

Test MCP endpoint

You can use the following endpoint to connect to the Test MCP server:

```
https://<loop_hostname>/test/mcp
```

MCP tools available in Test

Use these tools when you need to run tests, list tests in a branch, or fetch test outcomes and details. Do not use these tools for build orchestration, deployment execution, repository pull request actions, or planning updates.

Parameter conventions:

- `projectId` is required for all project-scoped operations.
- `branch` must be a valid branch name in the target project.
- `assetId` identifies the test asset to execute.
- `resultId` identifies an execution result for detailed lookup.
- Use IDs from prior list and get responses; do not infer IDs.

Tools	Description	Input	Output
testhub__get_projects	Retrieves a list of projects available in DevOps Test Hub.	None	List of available projects.

testhub__list_tests	Retrieves a list of tests in DevOps Test Hub for a specific project and branch.	projectId, branch	List of tests for the specified project and branch.
testhub__run_test	Runs a test in a specific project.	projectId, assetId, branch	Details of the executed test, including execution ID.
testhub__get_result_by_id	Retrieves the test run result and detailed result data.	projectId, resultId	Test execution result and detailed data.
testhub__get_results	Retrieves test result summaries for multiple executions in DevOps Test Hub.	projectId, branch	Test result summaries for multiple executions.

To configure MCP endpoint in VS Code, See Configuring an MCP server endpoint in VS Code on page 757.

Measure MCP endpoint and tools

Use Measure MCP tools to query value stream and engineering metrics, including workflows, deployments, builds, gate checks, and trend-based reporting.

MCP endpoint

You can use the following endpoint to connect to the Measure MCP server:

```
https://<loop_hostname>/velocity/mcp
```

MCP tools available in Measure

Use these tools when you need analytics, KPI trends, value stream reporting, or workflow health insights. Do not use these tools to modify planning records, execute builds, run deployments, run tests, or manage repository pull requests.

Parameter conventions:

- `tenantId` is required for tenant-scoped queries.
- `workflowId` is required for workflow-scoped metrics and statistics.
- `id` is the application identifier for application-by-ID queries.
- For date filters, use `YYYY-MM-DD` unless the endpoint requires ISO date-time.
- Use positive integers for pagination fields when supported.

Tools	Descriptions	Inputs	Output

Measure_applicationByIdQuery	Fetches application details for a given application ID or all applications for a tenant.	id, tenantId	Application details for the given ID or a list of all applications for the tenant.
Measure_workflowQuery	Retrieves detailed information for a specific Value Stream.	workflowId	Detailed information for the specific Value Stream/Workflow.
Measure_teamspacesForTenantQuery	Fetches teamspace information for a given tenant.	tenantId	List of teamspace information for the given tenant.
Measure_teamsForCurrentUserQuery	Fetches the tenant ID associated with the current user.	None	The tenant ID associated with the current user.
Measure_buildsQuery	Retrieves build metrics to analyze build success rates, commits, pull requests, and deployments against builds.	None	Build metrics and related data (success rates, commits, PRs, deployments).
Measure_workflowsForTenantQuery	Retrieves all workflows (VSM or value streams) and their details associated with a specific tenant.	tenantId	List of all workflows (VSM or value streams) and their associated details for the tenant.
Measure_metricsQuery	Retrieves metrics data based on specified filters and pagination options.	None	Metrics data based on the specified filters.
Measure_deploymentsQuery	Retrieves deployment data with related builds, issues, and pull requests to analyze release frequency.	None	Deployment data, including associated Builds, Issues, and PRs.
Measure_latestWorkflowGateChecksQuery	Retrieves the latest value stream workflow gate checks to verify whether quality gates are passing.	None	The latest workflow gate checks status.

Measure_workflowStatsByIdQuery	Retrieves key statistics for a specific workflow (value stream).	workflowId	Detailed key statistics for the specific workflow (value stream).

DevOps Loop MCP server configuration in VS Code

You can connect **Visual Studio Code** (VS Code) with GitHub Copilot to the DevOps Loop MCP server to run MCP-based queries that retrieve and update loop resources directly from VS Code. After the connection is established, the MCP server configuration is saved and remains available across VS Code sessions.

You can add the DevOps Loop MCP server in VS Code by using one of the following methods:

- By using the command palette. See Adding the DevOps Loop MCP server in VS Code by using the command palette on page 127.
- By editing the MCP configuration file. See Adding the DevOps Loop MCP server in VS Code by editing the MCP configuration file on page 128.

Adding the DevOps Loop MCP server in VS Code by using the command palette

You can use the VS Code command palette to quickly add the DevOps Loop MCP server without manually editing configuration files.

Before you begin

You must have completed the following tasks:

- Ensured that you have installed VS Code and the GitHub Copilot extension.
- You must have a valid MCP server endpoint URL. For example: `https://hostname/loop/mcp`

1. Open **VS Code**.
2. Open the **Command Palette**.
   - Windows/Linux: Ctrl+Shift+P
   - macOS: Cmd+Shift+P
3. Search and select **MCP: Add Server**.
4. Choose **HTTP** as the server type when prompted for the transport type.
   The DevOps Loop MCP server uses HTTP-based transport.
5. Enter the **MCP Server URL** in the **URL** field.
   VS Code automatically generates the required `mcp.json` configuration.
6. Enter a name for the server.

   For example: Loop MCP server.
7. Select the location to save the configuration:
   - **Global** – Makes the MCP server available in all VS Code workspaces.
   - **Workspace** - Makes the MCP server available only in the current workspace.

- **Workspace** – Makes the MCP server available only in the current folder/project.

  Note: Selecting **Workspace** automatically creates a `.vscode` folder and generates a `mcp.json` file inside it.

8. Select **Workspace** to save the configuration for the current workspace.
   VS Code creates a `.vscode` folder and generates an `mcp.json` file if they do not already exist.

Results

You have added the DevOps Loop MCP server to VS Code and is available in the MCP Servers view.

What to do next

After adding the DevOps Loop MCP server, you can view and manage the configured MCP server in VS Code. You can start the MCP server and complete the authentication process before using the MCP tools. See Viewing and managing MCP servers in VS Code on page 129.

Adding the DevOps Loop MCP server in VS Code by editing the MCP configuration file

You can manually configure the DevOps Loop MCP server by editing the `mcp.json` configuration file in your workspace.

Before you begin

You must have completed the following tasks:

- Ensured that you have installed VS Code and the GitHub Copilot extension.
- You must have a valid MCP server endpoint URL. For example: `https://hostname/loop/mcp`

1. Open **VS Code**.
2. Open any project in VS Code.
3. Create a new folder named `.vscode` at the root of the workspace, if not present.
4. Create a file named `mcp.json` inside the `.vscode` folder.
5. Paste the MCP configuration JSON. For example:

```
{
  "servers": {
    "loop-mcp": {
      "url": "https://hostname/loop/mcp",
      "type": "http"
    }
  },
  "inputs": []
}
```

6. Replace the `hostname` with your actual DevOps Loop server hostname.
7. Optional: To connect to a specific DevOps Loop capability, update the URL to the corresponding MCP endpoint.
   Each capability, such as Plan, Measure, Control, Build, Deploy, and Test, provides a dedicated MCP endpoint.
8. Save the `mcp.json` file.

Results

The DevOps Loop MCP server configuration is added to VS Code and is available in the MCP Servers view.

What to do next

After adding the DevOps Loop MCP server, you can view and manage the configured MCP server in VS Code. You can start the MCP server and complete the authentication process before using the MCP tools. See Viewing and managing MCP servers in VS Code on page 129.

Viewing and managing MCP servers in VS Code

After you add the DevOps Loop MCP server, you can view and manage the MCP server in VS Code.

Before you begin

You must add an MCP server in VS Code. See Adding the DevOps Loop MCP server in VS Code by using the command palette on page 127 or Adding the DevOps Loop MCP server in VS Code by editing the MCP configuration file on page 128.

1. Open VS Code.
2. Click the **Settings** icon in the chat interface.
   The **Agent Customizations** window is displayed.
3. Select **MCP Servers**.
   The configured MCP servers are displayed.
4. Right-click the MCP server you want to manage and select one of the following actions:
   - Click **Start Server** to start the MCP server.
   - Click **Stop Server** to stop the MCP server.
   - Click **Enable Server** to make the MCP server available for use.
   - Click **Disable Server** to prevent the MCP server from being used.
   - Click **Show Configuration** to view the configuration of the added MCP server.
5. When prompted, authenticate with the MCP server:
   a. Click **Allow** when VS Code displays a notification indicating that the MCP server requires authentication.
   b. Click **Open** when prompted to open an external website.
   c. Sign in to DevOps Loop by using your credentials.
   d. Review the consent request and click **Yes** to grant access.

   After successful authentication, the available MCP tools are displayed.

   Note: MCP server sessions expire after 30 minutes of inactivity. If no tool calls occur during this period, you must authenticate again before using the MCP server.

Results

You can view the status of configured MCP servers and manage their availability from the **MCP Servers** view. Authenticated MCP servers are available for use in VS Code.

Connecting Claude to the DevOps Loop MCP server

You can connect Claude to the DevOps Loop MCP Server and use it as a connector to retrieve and update loop resources.

Before you begin

You must have completed the following tasks:

- Ensured that Claude is installed and you have an active **Claude Pro subscription** (minimum).
- Obtained the MCP Server URL. For example: `https://hostname/loop/mcp`

1. Open the **Claude** application.
2. Navigate to **Settings > Connectors**.
3. Click **Add custom connector**.
   The **Add custom connector** window is displayed.
4. Enter the name of the connector. For example: Loop MCP server
5. Enter the MCP server URL in the **Remote MCP server URL** field.
6. Click **Add**.
   The MCP server is now added and listed on the **Connectors** page.
7. Click the **Connect** button next to the newly added Loop MCP server entry.

   Claude Desktop redirects you to the **DevOps Loop** login page.
8. Sign in with your DevOps Loop credentials.

   A consent screen is displayed requesting access to MCP tools, loop APIs (used internally for tool execution), and basic profile and role information.
9. Click **Yes** to approve and complete the connection.

   After successful authentication, Claude displays the list of available MCP tools provided by the DevOps Loop MCP Server. See Getting started with the DevOps Loop MCP server on page 112.

   Note: The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session will expire and you will be prompted to re-authenticate.

Results

You have successfully connected Claude to the DevOps Loop MCP Server.

Configuring an external MCP server in a teamspace

You can configure an external MCP server in DevOps Loop to connect to external MCP services and make them available to all loops in a teamspace.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Ensured that you must have a valid MCP server endpoint URL.

1. Navigate to **Settings** > **Installed Integrations**.
   The Installed Integrations page is displayed.
2. Click the **MCP** tab.
3. Click **Add Server**.
   The **Add Server** pane is displayed.
4. Enter a name to identify the MCP server in the **Name** field.
   For example, `GitHub MCP Server`.
5. Select the appropriate domain from the **Domain** list.

   Note: Each MCP server belongs to a Domain. Only one MCP server can be enabled for a Domain at a time. If you are adding a new MCP server to an existing domain, ensure that other MCP servers of the same domain are disabled.

   Note: If you select **Others** in the **Domain** list, the MCP server name is used as the domain value for that MCP server.

6. Enter the MCP server endpoint URL in the **Server URL** field.
7. Keep **OAuth** selected as the **Credential Type**.
8. **Optional**: Enter the client ID in the **Client ID** field.
9. **Optional**: Enter the client secret in the **Client Secret** field.
10. Click **Save**.

    The OAuth authentication window is displayed.

11. Complete the OAuth authentication process.
    The MCP server is added to DevOps Loop within a teamspace.
12. Use the toggle switch in the **Manage External MCP** list to enable the MCP server.

    Note: Newly added MCP servers are disabled, which is the default setting. If AI integration is inactive, the MCP server cannot be enabled.

Results

The MCP server is added to the Manage External MCP list. When the MCP server is enabled, it is available to all loops within the teamspace.

What to do next

The teamspace members can authenticate and use the MCP server through Loop Genie.

Teamspace management

A teamspace in HCL DevOps Loop is a secure, dedicated environment designed for multiple teams to work independently within a single shared DevOps Loop installation.

When you create a teamspace in DevOps Loop, the platform automatically provisions corresponding components for all integrated applications. Each component is provisioned with a corresponding name that reflects the Teamspace, ensuring consistency and seamless integration across all the applications.

The following table lists the configuration of each integrated application within the teamspace provisioning:

Application	Description
Plan	A new teamspace is created.
Control	An organization is created.
Code	A teamspace is created.
Deploy	A team is created.
Test	A teamspace is created.
Measure	A teamspace and a default team are created.

You can manage and switch between multiple teamspaces by clicking the **Switch Teamspace** icon . The following options are available to help you create and manage team spaces in DevOps Loop:

Action	Icon	Description
Create a teamspace		Select **Create TeamSpace** to create a team space within DevOps Loop.
Invite members to a teamspace		Select **Edit Teamspace** to add members to the teamspace.

These configurations provide your team with a fully integrated DevOps environment, streamlining collaboration, automation, and traceability across the software delivery lifecycle.

Related information

Creating a teamspace on page 132

Adding or inviting members to a teamspace on page 134

Joining a teamspace as an invited member on page 135

Creating a teamspace

You can create a teamspace in HCL DevOps Loop to provide your team with a dedicated, secure workspace that automatically spans across all the capabilities like Plan, Control, Code, Build, Test, Deploy, and Measure.

Before you begin

Ensure that the following prerequisites are completed:

- You are added to the **User** group in **Keycloak**.
- Ensure that you have administrative privileges in DevOps Loop.

1. Log in to DevOps Loop.
2. Click the **Teamspace** button in the left navigation panel.
3. Click **+ Create TeamSpace** in the **Switch Teamspace** panel.

   Result

   The Create Teamspace page is displayed along with the **Define Teamspace** and **Invite Members** tabs.
4. Perform the following steps in the **Define Teamspace** tab.
   a. Provide a name in **Enter TeamSpace Name**. The maximum character limit is 20.
   b. **Optional**: Provide a brief description of the teamspace in **Description**.
5. Click **Next**.

   Result

   The **Invite Members** tab is displayed.
6. Perform the following steps in the **Invite Members** tab.
   a. Click **Add External Members**.
   b. Search members with a valid email address or select them from the dropdown list and click **Add**.
   c. Type the email address of the members who are not in the Keycloak user directory and click **Add**.

   Note: You can enable or disable the teamspace access for the added members in a teamspace.

   - If the members already exist in the Keycloak user directory, they are directly added as members in the teamspace.
   - If the members do not exist in the Keycloak user directory, an invitation email is automatically triggered after the teamspace is created.

     Note: You can add multiple members to your invite list.
7. Click **Create TeamSpace**.

   The Teamspace setup progress page is displayed, showing each phase of the setup. DevOps Loop begins provisioning the TeamSpace across all integrated capabilities.

Note: A Control repository is added to Build, and a webhook is set up in Control to synchronize data between Control and Build.

If an error occurs during teamspace creation, an error message is displayed with options to **Retry** or **Cancel and retry later**. Click **Retry** to attempt the operation again. If the problem persists, check the tenant server pod logs and the logs for the associated product components to obtain additional information.

Note: The error badge indicates that the teamspace setup is incomplete or has configuration issues.

For a visual walkthrough of creating a teamspace watch the following video:
../videos/creating%20a%20teamspace.mp4

Results

You have created a teamspace, which is provisioned across all the capabilities of DevOps Loop. Additionally, the loop creation is enabled.

Related information

Adding or inviting members to a teamspace on page 134

Joining a teamspace as an invited member on page 135

Removing a member from a teamspace on page 135

Adding or inviting members to a teamspace

You can add existing members or invite new members to join a teamspace in DevOps Loop by using the **Edit Teamspace** option.

Before you begin

- You must be a teamspace owner.

1. Log in to DevOps Loop.
2. Select the teamspace in which you want to add members.
3. Click **Edit Teamspace**.

   Result

   The **Edit members** pane is displayed.
4. Perform the following steps in the **Invite Members** tab.
   a. Search members with a valid email address or select them from the dropdown list and click **Add**.
   b. Type the email address of the members who are not in the Keycloak user directory and click **Add**.
5. Click **Update Members**.

If the members already exist in the Keycloak user directory, they are directly added to the corresponding teamspaces across all integrated applications.
If the members do not exist in the Keycloak user directory, an invitation email is automatically triggered to join the teamspace.

Results

You have added members to your teamspace. If the members are not part of DevOps Loop, an invitation email is sent to them to join the teamspace.

Joining a teamspace as an invited member

You can join a teamspace in DevOps Loop when you receive an invitation from a teamspace owner. The invitation email includes a link to update your account and access the DevOps Loop.

Before you begin

You must have completed the following tasks:

- Confirmed that you have received an invitation email from DevOps Loop.
- Ensured that you are not signed in with another account in the same browser session.

1. Open the invitation email sent by DevOps Loop.
2. Click the **Link to account update** link in the email.
   The account update page is displayed in a new browser tab.

   Note: If you are already signed in with another DevOps Loop account in your default browser, open the link in a different browser or in an incognito or private window in your default browser.

3. Click **Click here to proceed**.
4. Type **New password** and **Confirm password** and click **Submit**.
5. Type the **Email id**, **First name**, and **Last name** on the account setup page.
6. Click **Submit**.
   You are added to the invited teamspace.

Related information

Teamspace management on page 132

Removing a member from a teamspace

You can remove a member from a teamspace when they no longer need access.

Before you begin

You must have completed the following tasks:

- Ensured that you have administrative privileges in DevOps Loop.
- Ensured that the member is removed from all associated loops before removing from the teamspace.

1. Log in to DevOps Loop.
2. Select the teamspace from which you want to remove a member.
3. Click **Edit Teamspace**.

   The **Edit Members** pane is displayed.

4. Click the Delete icon next to the member you want to remove in the Members List section.
5. Click**Update Members**.

Results

The member is removed from the teamspace.

Related information

Adding or inviting members to a teamspace on page 134

Uninstalling DevOps Loop

You can safely uninstall DevOps Loop from your environment, ensuring that all resources and data are properly removed.

Note: For demo setups and air-gapped environment, follow the same Kubernetes service uninstall procedures used for your platform.

Uninstalling DevOps Loop from IBM Cloud Kubernetes Service (IKS)

To reinstall HCL DevOps Loop when an ongoing installation fails, you can uninstall DevOps Loop and its components from theIBM Cloud Kubernetes Service (IKS) cluster.

Before you begin

You must have completed the following tasks:

- Installed DevOps Loop.
- Closed DevOps Loop, any open web browsers, and all other applications that are enabled by DevOps Loop.
- Ensured that, if applicable, existing namespaces used by Code dev containers running in user namespaces are deleted before starting the upgrade.
- **Optional**: Backed up data from the previous version of DevOps Loop.

1. Navigate to **Cluster Management > Clusters > Overview > Actions > Connect via CLI** in your IBM Cloud account and switch context to your cluster.
2. Run the following command to uninstall DevOps Loop:

   ```
   helm uninstall $HELM_NAME -n $NAMESPACE
   ```

   Where,

   ```
   NAMESPACE=devops-loop
   HELM_NAME=devops-loop
   ```

   The PersistentVolumeClaims and PersistentVolumes that were created during the installation are not deleted automatically. If you reinstall DevOps Loop, the user data is reused unless you specifically delete those volumes.
3. Run the following command to delete everything, including the user data contained in claims and persistent volumes:

   ```
   kubectl delete namespace $NAMESPACE
   ```

Results

You have uninstalled DevOps Loop from the IKS cluster.

Uninstalling DevOps Loop from Kubernetes Service (K8S)

To reinstall HCL DevOps Loop when an ongoing installation fails, you can uninstall DevOps Loop and its components from the Kubernetes Service (K8S) cluster.

Before you begin

You must have completed the following tasks:

- Installed DevOps Loop.
- Closed DevOps Loop, any open web browsers, and all other applications that are enabled by DevOps Loop.
- Ensured that, if applicable, existing namespaces used by Code dev containers running in user namespaces are deleted before starting the uninstall.

1. Switch to your K8S cluster by using the following command:

   ```
   kubectl config use-context <context-name>
   ```
2. Run the following command to uninstall DevOps Loop:

   ```
   helm uninstall $HELM_NAME -n $NAMESPACE
   ```

   Where,

   ```
   NAMESPACE=devops-loop
   HELM_NAME=devops-loop
   ```

   Note: User-specific DevContainer namespaces are not removed automatically when DevOps Loop is uninstalled. These namespaces are created outside of the DevOps Loop Helm release and therefore are not deleted when the Helm release is removed.

After uninstalling DevOps Loop, review the cluster for any DevContainer namespaces that are no longer required and delete them manually:

```
kubectl delete namespace <devcontainerNamespace>
```

DevContainer namespace names are derived from the user's email address, with special characters converted to Kubernetes-compatible character sequences. For example, the email address `xyz@example.com` is converted to the namespace `xyz-40-example-2e-com`.

The PersistentVolumeClaims and PersistentVolumes that were created during the installation are not deleted automatically. If you reinstall DevOps Loop, the user data is reused unless you specifically delete those volumes.

3. Run the following command to delete everything, including the user data contained in claims and persistent volumes:

```
kubectl delete namespace $NAMESPACE
```

Results

You have uninstalled DevOps Loop from the K8S cluster.

Chapter 5. Working with Loops

A loop is a workflow that connects DevOps activities of your team from planning and development through build, deployment, testing, and release phases.

You can find information about creating and managing loops, learning loop, managing members, and using dashboards to monitor progress across all capabilities of DevOps Loop.

Loop management

The loop consists of several interconnected phases that ensure continuous software development, testing, deployment, and monitoring.

Loop creation ensures seamless integration between development and operations, leading to faster and more reliable DevOps workflow. The following changes take place as a result of loop creation:

- Provisions your planning, source control, testing, deployment, and metrics tools for a ready-to-use project environment.
- Connects all capabilities by installing and configuring essential plugins, webhooks, repositories, and integrations for real-time data sharing and visibility.
- Provides access to a unified dashboard and links for direct access to each application.
- Adds users in each capability and sets specific access rights across all applications.

Overall, loop creation reduces the time and manual effort required to configure and maintain your development toolchain so that you can focus on innovation.

You can explore loop capabilities by enabling Learning Loop option. This allows you to learn and practice workflows by working with sample projects, work items, and configurations without starting from an empty setup. See Learning Loop overview on page 151.

Managing loops

You can use the following options on the **Home** page:

Actions	Description
View all the loops	The **All loops** tab lists all the loops within your teamspace.
View the recently added loops	The **Recently Added** tab lists the recently added loops within your teamspace.
View My favorites	The **My Favorites** tab lists the loops that are marked as favorite within your teamspace.
View Disabled loops	The **Disabled** tab lists the loops that are disabled within your teamspace.
View the incomplete loops	The **Incomplete** tab lists all the incomplete loops within your teamspace. You can click **Retry** to complete the creation of the loop.

Actions	Description
Search loops	You can search for a loop by clicking the icon and entering the name in the **Search Loops** field when the list of loops is lengthy.
Create a loop	You can create a new loop by clicking the **Create Loop** button.
Edit a loop	You can modify the list of team members who can access the loop by clicking the **Settings** icon .
Mark as favorite	You can mark a loop as a favorite to make it easier to find by clicking the **Add to favorites** icon .
Open Loop Genie	You can interact with Loop Genie and send your queries related to the loop by clicking the button.
View dashboards	You can view the overview and detailed dashboards for the loop by clicking
View loop details	You can view loop details such as configuration details and its linked resources, such as test projects and control repositories, by clicking the **Loop Details** option on the loop.

Creating a loop

You can create a loop to build a pipeline of tasks across the integrated capabilities in HCL DevOps Loop.

Before you begin

Ensure that the following prerequisites are completed:

- You are added to the **User** group in **Keycloak**.
- Created a teamspace or a member of an existing teamspace.
- Read and understood the information in About user roles and access permissions on page 83.

About this task

When you create a loop, DevOps Loop creates specific resources in each of the applications and also, installs and configures the required plugins. During the loop creation, when the plugins are installed and configured, personal access tokens are created in **Plan**, **Control**, and **Test**. On loop creation, the respective users are added to the specific resources in each of the capabilities.

1. Click **Teamspace > <teamspace_name>** to navigate to your teamspace.
2. Click **Create Loop** on the home page of your teamspace.
   The **Define Loop** tab is displayed.
3. Perform the following actions in the **Define Loop** tab:

a. Enter a loop name.

> **Note:** The first five characters for the loop name must always be unique, and they should not be reused while creating another loop.

b. **Optional**: Enter details about the loop in **Description**.
c. **Optional**: Enable the **Learning Loop** option to prepopulate the loop with demo data for exploring workflows and practicing tasks.

4. Click **Next**.
   The **Invite Members** tab is displayed.
5. **Optional**: Perform the following actions in **Invite Members** tab.
   a. Click Add External Members.
   b. Enter the email ID of the team member to be added to the loop in **Search members**.
   c. Click **Add**.
6. Click **Next**.
   The loop setup process begins. A progress view displays the status of each capability being initialized. Once all the capabilities are fully initialized, the loop is successfully created.
   For details about the capabilities and resources configured during loop creation, see Capabilities and resources configured during loop creation on page 142.

   What to do next: After the loop is created, DevOps Loop automatically provisions the required resources and integrations across all capabilities. You can begin using these resources based on your workflow needs.

   The following examples show common actions you may perform after creating a loop. These are **optional** and depend on the tasks you want to complete within the loop:
   - View the Dashboard to see summaries and statuses of the activities through visualizations. See Dashboards and insights on page 154.
   - Create components and applications in DevOps Deploy. See Creating components in Deploy for a loop and Creating applications in Deploy for a loop.
   - For Control integration, create a file named `input.txt` under the repository in Control.

   For a visual walkthrough of creating a loop watch the following video:
   ../videos/creating%20a%20loop.mp4

Related reference

REST commands for Deploy

Related information

#unique_188

Capabilities and resources configured during loop creation

When you create a loop, the platform automatically provisions resources, installs and configures plugins, generates tokens, establishes integrations, and assigns appropriate permissions to the loop creator and invited members across multiple capabilities.

Capability-specific resources and configurations

Capability or Plugin	Configuration during loop creation
Plan	• Creates an application in the same teamspace where the loop is created. • Sets up email notifications for work item changes. • Names the application using the first five characters of the loop name.
Control	• Creates a repository with the same name as the loop in the organization associated with the teamspace.
Code	• No resources are created at this stage.

Build	• Creates a team with the following convention: `<TeamspaceName~LoopName>`.
	• Creates the following templates:
	  ◦ Project
	  ◦ Process
	  ◦ Source
	  ◦ Jobs
	• Creates a job template with basic CI/CD steps.
	• Creates the **Deploy** and **Control** integrations.
	• Creates the main project with the naming convention **TeamSpace~Loop:LoopName** (The UI displays this convention as **TeamSpace~Loop**) and configures it using the Process and Source templates.
	 Note: A project is associated with only one loop.

Test	Creates a project with the same name as the loop within the teamspace.

Capability or Plugin	Configuration during loop creation
Deploy	• Creates an application with the same name as the loop and a team named <TeamspaceName~LoopName>. • Creates three empty environments: Development, QA, and Production. • Creates a template pipeline and an empty application process.
Plan Plugin	• Installs and configures the Plan plugin in Measure. • Installation occurs once; configuration occurs with each loop. • Generates a personal access token in Plan which is used to configure the Plan Plugin for Plan-Measure connection. • Auto-synchronizes every 5 minutes. • Default version: 1.2.7
Build Plugin	• Installs and configures the Build plugin in Measure. • Installation occurs once; configuration occurs with each loop. • Auto-synchronizes every 5 minutes. • Default version: 1.1.5
Control Plugin	• Installs and configures the Control plugin in Measure. • Installation occurs once; configuration occurs with each loop. • Generates a personal access token in Control which is used to configure the Control Plugin for Control-Measure connection. • Auto-synchronizes every 5 minutes. • Default version: 1.1.3
Deploy Plugin	• Installs and configures the Deploy plugin in Measure during the first loop creation. • Generates a personal access token in Deploy which is used to configure the Deploy Plugin for Deploy-Measure connection. • Auto-synchronizes every 5 minutes. • Default version: 5.1.1
Test Plugin	• Installs and configures the Test plugin in Measure. • Installation occurs once; configuration occurs with each loop.

Capability or Plugin	Configuration during loop creation
	• Generates an offline user token in Test which is used to configure the Test Plugin for Test-Measure connection. • No auto-synchronization. • Default version: 1.0.32
Control–Plan Integration	• Links repository changes (pull requests, commits, branch creation) in Control to work items in Plan. • Pull request titles and branch names should start with the work item ID for proper syncing. • Changes appear in the SCM Event section of the work item in Plan. • A personal access token in Plan configures a webhook in Control.
Test–Plan Integration	• Creates work items in Plan from Test Hub via integration. • A personal access token in Plan is used for setup. • Verify by opening your Test Hub project and navigating to Manage Integration > Change Management.
Test–Control Integration	• Maps the repository created in Control to the Test Hub project associated with the loop. • View it in Test Hub under Manage Configuration > Repositories. • Enables Test Hub to pull test assets from Control.
Measure	• Creates a team <TeamspaceName~LoopName Team> and adds the creator/admin with full permissions. • Creates a value stream and pipeline associated with the team, using application data from Deploy by using the YAML plugin in Measure.
Analytics	• Creates dashboards in Opensearch for the newly created loop.
Loop Genie	• Creates a repository in Control with the naming convention `<loop_name>~TASKS`. • PredefinedTask files are uploaded in the TASKS repository.

Capability or Plugin	Configuration during loop creation
Branch Protection	• Configures branch protection on the Control repository's main branch

Access and Permissions

The loop creator is granted the highest permissions for all resources in the integrated capabilities. Invited users are given only basic permissions by default when they join a loop. They can manually elevate their own permissions through the UI for any resource as needed.

Viewing loop details

You can use the **Loop Details** page to view a complete overview of a loop, including its key configuration and resource information.

When you want to view a loop's configuration details and its linked resources, such as test projects and control repositories, select the **Loop Details** option on the loop. Each resource includes metadata such as its name, URL, and creation date, making it easy to understand how the loop is structured and which assets it relies on.

From the **Loop Details** page, you can add additional test projects and control repositories as needed, so that you can expand or update the loop's configuration without leaving the page.

The page also includes a team members section. Here, you can view all the members who are part of the loop, along with their role (Admin or Team Member), their email ID, and the other loops they belong to. You can search through the list of members or make updates using the available edit capabilities, helping you keep membership aligned with your project requirements.

Related information

Adding members to a loop on page 145

Adding members to a loop

You can add members to a loop who are already part of your teamspace so they can collaborate and stay updated within it.

Before you begin

You must have completed the following tasks:

- Created a loop.
- Ensured that you have administrative privileges in DevOps Loop.
- Ensured that the member is already added to the teamspace.

1. Log in to DevOps Loop.
2. Click **Teamspace > <teamspace_name>** to navigate to your teamspace.
3. Click **Loop Details** option in the Loop card for which you want to add members.

The **<Loop name> Details** page is displayed.

4. Navigate to **Team Members** tab and click **Edit Members** button.
   The **Edit Members** pane and list of members in your teamspace is displayed.
5. Type the email address in the **Search** bar or select the member from the list and click **Add**.
6. Click **Update Members**.

Results

The member is added to the loop and its capabilities.

Related information

Removing a member from a loop on page 146

Removing a member from a loop

You can remove a member from a loop when they no longer need access.

Before you begin

You must have completed the following tasks:

- Created a loop.
- Ensured that you have administrative privileges in DevOps Loop.

About this task

When you remove a member from a loop, the member is also removed from all connected capabilities within that loop.

1. Log in to DevOps Loop.
2. Select the loop from which you want to remove the member on the **All Loops** tab.
3. Click the Settings icon and select **Edit**.

   The **Loop Members** pane is displayed.

4. Select the member in the **Members List** section, and click the **Delete** icon .
5. Click **Save**.
   The member is removed from the loop and its capabilities.

Related information

Adding members to a loop on page 145

Disabling a loop

You can disable a loop to pause its activities and stop updates. You can enable it again from the **Disabled** tab whenever you need it.

Before you begin

You must have completed the following tasks:

- Created a loop.
- Ensured that you have administrative privileges in DevOps Loop.

About this task

When you disable a loop, the following connected capabilities within that loop are impacted:

- **Plan**: Emails, alerts, and search functions for work item updates are disabled.
- **Control**: The Plan-Control webhook is deactivated and all the information in the events (such as pull request, branch creation etc) stop flowing from Control to Plan.
- **Measure**: Data flow from Control, Plan, Test and Build is interrupted and therefore value streams and dashboards no longer receive updates.

1. Log in to DevOps Loop.
2. Navigate to the loop that you want to disable from the **All Loops** tab.
3. Click the Settings icon in the loop which you want to disable
4. Select **Archive Loop** option.

   The loop is disabled and moved to the **Disabled** tab.

Related information

Enabling a loop on page 147

Enabling a loop

You can enable a loop to restore its activities and allow updates to flow again.

Before you begin

You must have completed the following tasks:

- Created a loop.
- Disabled the loop you want to enable.
- Ensured that you have administrative privileges in DevOps Loop.

About this task

When you enable a loop, the following connected capabilities are restored:

- **Plan** – Emails, alerts, and search functions for work item updates are re-enabled.
- **Control** – The webhook is activated and all the events such pull request, branch creation etc starts flowing from Control to Plan.
- **Measure** – Data flow from **Control**, **Plan**, **Test** and **Build** resumes, and therefore value streams and dashboards receive updates.

1. Log in to DevOps Loop.
2. Select the Loop that you want to enable from the **Disabled** tab.
3. Click the Settings icon and select **Enable**.

   The loop is enabled and appears in the **All Loops** tab.

Related information

Disabling a loop on page 147

Loop Segments overview

When a loop contains multiple microservices, you can add loop segments to manage each microservice independently. Each segment creates a dedicated set of resources and integrations for that microservice.

These resources support the development lifecycle of the microservice and can include items such as:

- A dedicated control repository
- A separate plan component
- Supporting test and build integrations

Each segment behaves like a sub-loop and provides capabilities that are specific to a microservice.

A segement:

- Supports its own build and validation flow
- Tracks changes specific to the microservice
- Remains connected to the overall application lifecycle

You can use segments when you are working with multiple microservices in a single application, each microservice needs separate repositories, builds or testing processes. Segments help maintain clear separation between microservices without creating multiple loops for each microservice.

For example: You build an e-commerce application using a microservices architecture, such as frontend service, order service, and payment service. All these microservices belong to the same application and share a common delivery lifecycle, so you create one Loop to represent the overall application. However, you create a **Segment** for each microservice inside the parent loop. For example, you create a segment named **Order Service**.

A loop can contain multiple segments and each segment contributes to the overall value stream.

Related information

Adding a loop segment on page 149

Adding a loop segment

You can add a loop segment within a loop to manage a microservice that requires its own repositories, builds, and integrations.

Before you begin

Ensure that the following tasks are completed:

- Created a teamspace or a member of an existing teamspace.
- Created a loop.

About this task

Segments are used when an application contains multiple microservices that require independent repositories, builds, and testing processes. Each segment represents a microservice within the loop.

1. Click **Loop Details** option in the loop card for which you want to add a loop segment.
2. Click **Add connection > Loop segment** in the **Loop Connections** tab.
   The **Add Loop Segment** panel is displayed.
3. Enter a Segment name in **Name** field.

   Segment name must be at least 3 characters long.
4. **Optional:** Enter additional details in **Description**.
5. Click **Add Segment**.

   The system begins the segment setup process and displays a progress view that shows the status of each initialization task. During this process, the system performs the following actions:

Figure 1. Capability-specific resources and configurations created in segment

Capability or Plugin	Configuration during segment creation
Plan	◦ Creates a plan component.
Control	◦ Creates a repository with the same name as the segment name in the organization associated with the teamspace. ◦ Applies branch protection rules to the repository.
Build	◦ Creates a project with the naming convention **TeamSpace~Loop:LoopName:SegmentName** and configures it using the Process and Source templates. **Note:** A project is associated with only one loop or segment. ◦ Creates a job template with basic CI/CD steps for the segment. ◦ The segment process is added as dependency within the process of the main project of the loop. ◦ The main project build is trigerred only when we have at least one version built for all the segments in the loop.
Test	◦ Creates a project with the name as **LoopName-SegmentName**within the teamspace. ◦ Configures Control repository in the created project. ◦ Configures webhook to upload test results to measure. ◦ Configures Plan integration in the created project.
Deploy	◦ Creates a component in Deploy with the name as **LoopName-SegmentName** within the application.
Build Plugin	◦ Updates the Build plugin configuration in Measure to add the newly created project for the segment.
Control Plugin	◦ Updates the Control plugin configuration in Measure to add the newly created repository for the segment.
Control–Plan Integration	◦ Links repository changes (pull requests, commits, branch creation) in Control to work items in Plan. ◦ Pull request titles and branch names should start with the work item ID for proper syncing. ◦ Changes appear in the SCM Event section of the work item in Plan. ◦ A personal access token in Plan configures a webhook in Control.

After all tasks are completed, the segment is successfully created and appears under the **LoopName Loop Segments** section in the **Loop Connections** tab.

Related reference

Capabilities and resources configured during loop creation on page 142

Learning Loop overview

Learning Loop provides preloaded sample data across multiple DevOps Loop capabilities. It helps you learn and practice workflows by exploring sample projects, work items, repositories, build pipelines, deployments, and tests without starting from an empty setup.

Learning Loops use a sample application that consists of frontend and backend microservices. The application components are organized into segments, which contain the source code, build configurations, deployment configurations, and test assets required to demonstrate an end-to-end DevOps workflow.

The Learning Loop, with its preloaded sample data, helps you explore and understand the following DevOps Loop features:

- Understand how events flow between capabilities. For example, a commit in Control triggers a build in Build and a deployment in Deploy.
- View a consistent end-to-end implementation of DevOps Loop capabilities for demonstrations, evaluations, and training.
- Explore how DevOps Loop manages multi-component applications through segments.
- Experiment with features and configurations in a non-production environment.

Learning Loops include an automated deployment experience for a sample application. Code changes flow through the complete pipeline from source control to build, deployment, and testing. The sample application consists of frontend and backend services that are managed independently through segments while remaining connected within the same Learning Loop.

Segments in Learning Loops

A segment is a sub-loop that represents an individual application component within a Learning Loop.

Learning Loops use segments to organize and manage multi-component applications. When a Learning Loop is created, DevOps Loop automatically provisions separate frontend and backend segments.

Each segment contains the following resources:

- Source code repository
- Build configuration
- Deployment configuration
- Test assets

Segments help you manage and deploy application components independently while maintaining visibility across the complete application lifecycle.

Scope

A Learning Loop creates resources across the following DevOps Loop capabilities:

- **Plan** - Creates a project with releases, sprints, and work items.
- **Control** - Creates and populates frontend and backend source code repositories with sample application code and deployment artifacts.
- **Build** - Creates projects, processes, and jobs for frontend and backend segments. Requires a manual step to configure an active agent.
- **Deploy** - Creates deployment resources and processes for frontend and backend segments. Backend services are deployed to Kubernetes, while frontend artifacts are published through the Learning Webserver.

Key characteristics

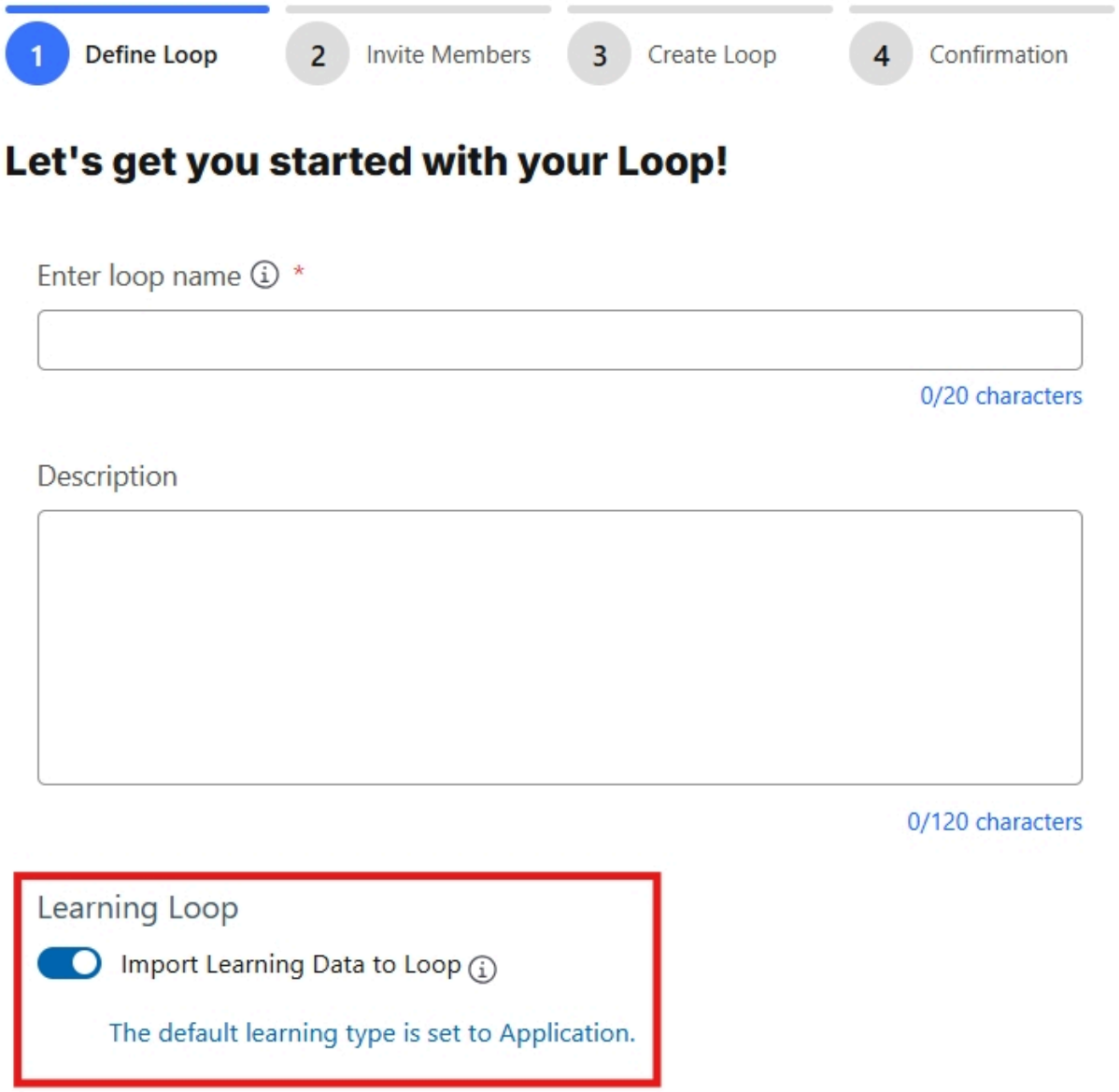
- Built for learning and exploration, not for use in production environments.
- Some manual steps, such as configuring build agents, are required.
- After the configuration, the end-to-end workflow is automated across Control, Build, Deploy, and Test.
- After the sample data is loaded, you can edit or extend it to fit your own configurations.
- Learning Loops support multi-component application architectures through frontend and backend segments.
- The frontend application is hosted through the Learning Webserver and is accessible through a unique URL generated for each Learning Loop.
- Each Learning Loop receives its own backend deployment and temporary database instance.

Note: The temporary database is not persistent and is reset if the backend deployment is restarted.

Creating a Loop with preloaded example data

When you want to explore DevOps Loop capabilities and learn how workflows operate across Plan, Control, Build, Deploy, and Test, you can use the Learning Loop option to preload sample data with pre-configured projects, repositories, build pipelines, deployments, and test assets.

1. Log in to DevOps Loop.
2. Click **Create Loop**.
3. Enter the required loop details, such as the **Loop Name** and **Description**.
4. Enable the **Learning Loop** option.



When you select the **Learning Loop** option, DevOps Loop automatically provisions a sample application and the associated resources and configurations across the Plan, Control, Build, Deploy, and Test capabilities.

The following resources and configurations are created:

- Frontend and backend segments.
- Source code repositories populated with sample application code.
- Build configurations for frontend and backend services.
- Deployment configurations for frontend and backend services.
- A hosted frontend application.
- A unique URL for accessing the Learning Loop application.

5. Navigate to **Test** and execute the test manually.
   The test result is displayed in the **Execution** tab.

Results

You have successfully created a Learning Loop with a sample application, pre-configured frontend and backend segments, and sample data across DevOps Loop capabilities.

Dashboards and insights

The dashboard consolidates data from all capabilities, such as Plan, Code, Control, Build, Test, Release, and Deploy, by offering a centralized view of the status. Dashboards are available with both summary and detailed perspectives, and they filter information specific to each loop.

After you created a loop, five dashboards for each subscription tier are automatically provisioned and linked directly to the workspace of each loop. These dashboards are universally visible, irrespective of the user's licensing tier. You can see tier-specific differences in the dashboard because of the depth and variety of data sources through integrated applications for a specific tier. The default dashboards are preconfigured and available for immediate use.

For advanced analytics, DevOps Loop includes a milestone risk assessment feature, which requires an optional plug-in.

Dashboard types

You can view the following dashboards for each loop in your teamspace:

Table 4. Available Dashboards in a loop

Dashboards	Description
Overview	The **Overview** dashboard provides a high-level summary of loop activity, including issues, team workload, test results, builds, and deployments.
Detailed	The **Detailed** dashboard provides more granular insights, such as issues, pull requests and commits, builds, deployments, test metrics, and timeline performances.
Delivery	The **Delivery** dashboard provides information about Lead Time, Cycle Time, Dev Cycle Time, Deployment Counts, Throughput, Distribution, Deployment Frequency, and Lead Time to Change.
Quality and Security	The **Quality and Security** dashboard provides the information about Functional Tests, Unit Tests, and API Tests.

Data authorization

All the dashboard data is displayed based on your loop membership and access rights. You can only see information for loops in which you are a member, so that sensitive data remains secure and collaboration across teams stays controlled.

Viewing dashboards

You can view the dashboards for the loops in your teamspace. The dashboards display key metrics such as issues, workloads, builds, deployments, milestones, lead time, and cycle time.

Before you begin

You must be a member of the loop.

About this task

You can view the data only for the loops you are a member of, and the information displayed is restricted based on your access rights.

1. Click the **Dashboards** tab in the navigation panel.
   The **Dashboards** page is displayed.
2. Select the dashboard from the **Dashboard name** column.

Results

The selected dashboard with metrics and data is displayed.

Related information

Editing dashboards on page 155

Dashboard types on page 154

Editing dashboards

You can customize the dashboard view by applying filters and selecting specific metrics such as test results, build history, or deployment status to focus on the data most relevant to your loop.

Before you begin

You must have completed the following tasks:

- Created a loop.
- Ensured you have administrative privileges.

1. Click the **Dashboards** tab.
   Result
   The **Dashboards** page is displayed.
2. Click the **Edit** icon .
3. Use any of the following options to edit the dashboard:

Option	Description
Full Screen	Expands the dashboard to full-screen mode for easier viewing.
Share	Shares the dashboard view with other users in the following format • Embed code • Permlinks
Clone	Creates a copy of the selected widget within the dashboard layout.
Swap	Allows for swapping the position of widgets within the dashboard layout.
Add	Adds new panels or widgets to the dashboard.
Cancel	Cancels the current edit operation.
User Margin Between Panels	Adjusts spacing between panels in the dashboard.
Show Panel Titles	Toggles panel titles on or off.
Add Filters	Applies a filter to the data to be displayed in the dashboard.

4. Click **Save** to save the changes.
5. Click **Refresh** to update the dashboard view with the latest data.

Results

You have customized the dashboard to focus on the metrics and information that are most important to your loop.

AI-powered search

When you want to search work items, records, and related data in DevOps Loop, you can use the search bar to locate information across multiple capabilities. You can search across a wide set of artifact types, including issues, builds, tool requests, commits, and deployments, and you can filter results by loop or by category to quickly narrow your search.

You can search for epics, defects, and stories across Plan, and the results are presented as interactive cards that include metadata. From these cards, you can directly access associated URLs for pull requests (PRs), commits, and builds, making it easy to move from search results to action.

You can generate AI summaries of search results to quickly understand the context of work items, issues, and related data. These summaries are generated for the top five results on each page, helping accelerate decision-making by providing quick insights as you navigate through your results.

Note: AI summaries are available only if the OpenAI or watsonx integration is enabled in DevOps Loop. If neither integration is enabled, DevOps Loop will not generate summaries.

Search results are automatically filtered based on your access rights. You will only see documents and records associated with loops of which you are a member, ensuring that sensitive data remains secure and collaboration across teams is controlled. This ensures that results are fully scoped to the current user, preventing visibility into team spaces you do not belong to.

Chapter 6. Capabilities of DevOps Loop

HCL DevOps Loop provides an integrated suite of capabilities such as Plan, Code, Control, Build, Test, Release, Deploy, Measure and Harbor (if enabled) to manage every stage of the software development lifecycle within a connected, continuous loop.

Each capability is designed for a specific purpose, but together they form a continuous feedback cycle, helping teams plan, build, deploy, and measure value efficiently.

Plan

Plan is a change management software application within the DevOps Loop that supports seamless ticket creation, tracking, and comprehensive issue management. With customizable workflow automation, it helps teams to streamline and adapt processes to meet their unique business requirements while preparing work for the next stages of the loop.

For more details refer to the Plan Documentation.

Code implementation with AI

The Code Genie feature provides an end-to-end integration within the Loop environment, allowing you to generate source code directly from a work item. By coordinating between DevOps Plan, Code, and Control products, Code Genie automates the transition from a descriptive task to a functional Pull Request (PR).

Before you begin

Code Genie leverages AI to translate work item details into actionable coding instructions. It analyses the Description and Conversation (comments) fields of a work item to determine the requirements for code generation. The tool then triggers a background process in which an AI-powered dev container applies the requested changes and submits them as a PR in the designated Control repository.

Before you begin, ensure that you have:

- AI requirement: You must have an AI integration enabled to use the Generate function for coding instructions. For more information, see the *Artificial Intelligence integration* topic in Plan Documentation.
- CLI token: For a CLI token, see Manage Coding CLI on page 215.
- Branching strategy: Use a branching strategy on page 233 with a protected main branch.
- Processing time: Code generation is not instantaneous. After clicking Implement in DevOps Plan, it might take some time for the PR to appear in Control.
- Manual review required: While Code Genie generates the code, ensure that a reviewer must still manually review and merge the PR to ensure quality and security.
- Branch protection: Standard branch protection rules in Control still apply, preventing unauthorized code from being pushed directly to protected branches.
- Unsaved instructions: If you close the slide-out panel before clicking Implement, any manually entered or AI-generated instructions that have not been submitted will be lost.

About this task

1. Open the work item: Navigate to the desired work item in DevOps Plan in full view mode.
2. Launch Code Genie: Click the Code Genie button located in the record toolbar.
3. Select repository: In the slide-out panel, select the appropriate Control repository from the dropdown menu. This action automatically loads the relevant dev container and coding guidelines.
4. Prepare coding instructions:
    - AI generation: Click **Generate** to let the AI create instructions based on the work item's description and comments.
    - Manual entry: Alternatively, type specific instructions or prompts directly into the text area.
5. Implement: Click the **Implement** button to initiate the background generation process.
6. Verify the PR: Navigate to the Control UI to view the automatically generated PR. You can review the files changed and any automated security scans that are run by Trivy.
7. Monitor in DevOps Plan: Return to the work item in the DevOps Plan and check the SCM events tab to verify that the PR is reflected there.

Results

Upon successful completion, Code Genie generates a complete PR in the linked Control repository. This PR includes the implemented code files based on your instructions and is ready for manual review and merging by assigned reviewers.

Code

Code is a cloud-based browser IDE on page 180 in a dev container on page 161 that is part of HCL DevOps Loop, allowing developers to write, compile, build, and debug code directly in the browser without requiring local setup.

Built on Visual Studio Code, Code offers pre-configured tools, extensions, and libraries, along with support for custom extensions. It includes a file system, terminal, and tools for debugging and testing while integrating with remote source control repositories for secure version control.

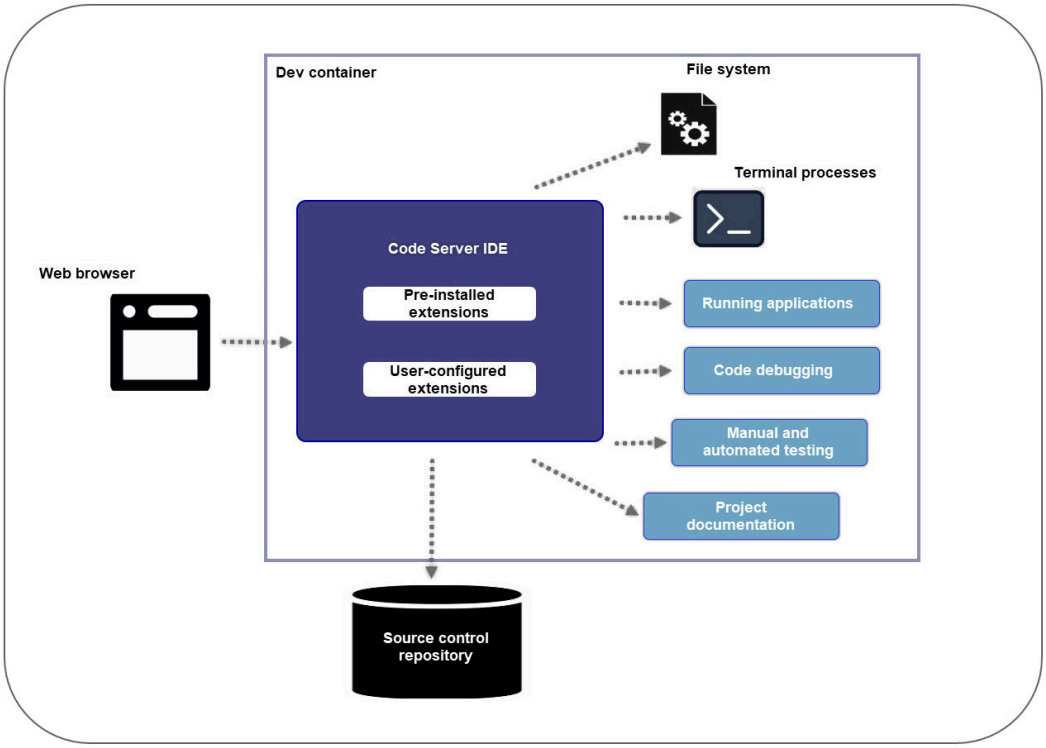
Continue reading to learn how Code enhances your development experience and integrates with DevOps Loop.

Code overview

Code is a cloud-based integrated development environment on HCL DevOps Loop, with which you can write, compile, build, and debug software applications in a so-called **dev container**, accessible directly from a web browser. The editor provides a completely configured development environment with preinstalled extensions, tools, and libraries. You can start coding without the need to set up the environment on your local machine.

Architecture

The general architecture of the Code platform is shown in the following image.



Code provides personal **dev containers** that are configured for members of a teamspace. By using each dev container, you can write, build, and debug your code in an integrated environment without affecting the work of other users in the team. Each container is isolated for a specific user to prevent conflicts with different project setups. Additionally, you can also customize the individual dev container to meet your specific needs, by ensuring that all the necessary extensions and libraries are available for use.

The dev container integrates essential tools and processes, with the web browser serving as an interface for accessing the browser IDE on page 180, supporting coding, debugging, version control, and more.

Visual Studio Code-based environment

The browser development environment is based on Visual Studio Code, which means that almost all features the latter provides are also available in Code. For more information, refer to Visual Studio Code documentation.

The development environment includes preinstalled extensions on page 181. However, you can also add custom extensions to suit project needs. It includes a file system for file management, a terminal for command-line operations, and tools for application execution, debugging, testing, and maintaining documentation.

Source control integration

Code integrates with remote source control repositories, ensuring secure, accessible version control and project continuity across sessions. Code offers built-in support for Git repositories (for example, repositories stored in https://help.hcl-

software.com/devops/plan/3.0.3/oxy_ex-1/control/content/index.en-us.html). Other types of source control repositories are supported by means of extensions that you can install into the dev container.

Accessing Code

To learn how to access the Code feature, see DevOps Loop user interface on page 30.

Related information

Git authentication for dev containers on page 191

Accessing dev containers

You can switch to your teamspace to access team-specific dev containers and their pre-configured settings.

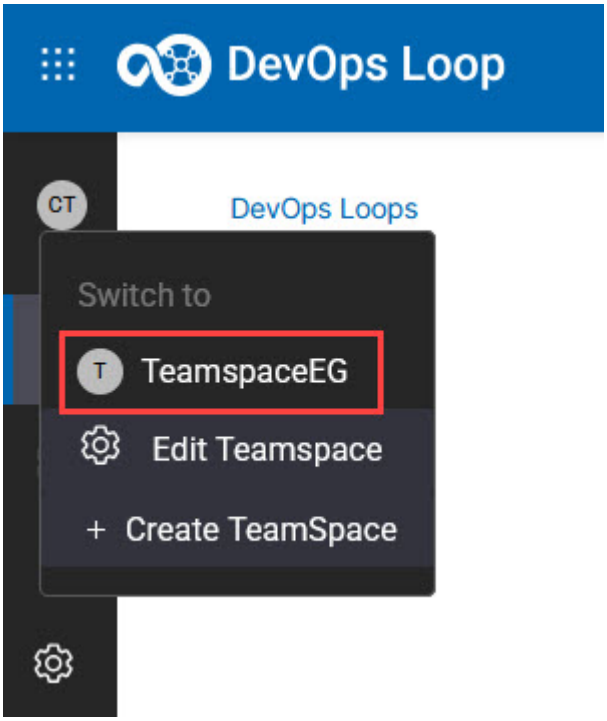
About this task

Upon login, you can navigate using two tabs: **Running Dev Containers** and **Other Dev Containers**.

If you have an assigned teamspace, you can launch preconfigured dev containers - such as Starter, Java, Python, or C++ - directly from the **Other Dev Containers** tab. Once launched, you can manage these running dev containers in the **Running Dev Containers** tab. You can find full details in the Working with dev containers on page 169 guide.

To access your team-specific dev containers:

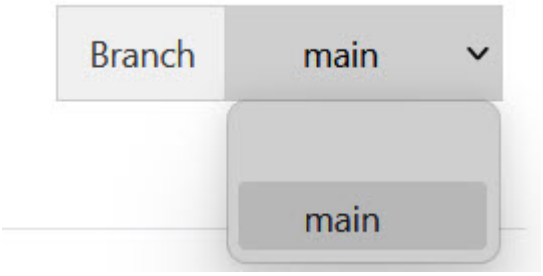
1. Click the switch teamspace button in the top-left corner.



2. Select the appropriate teamspace created by your team admin.

Note: If you do not see any teamspace listed, it means you have not been added yet. Contact your team admin to request access to a teamspace.

You can also switch to a different branch (other than main) using the Branch menu in the upper right corner of the IDE. The branch switch menu is visible only in **Other Dev Containers** tab. This allows for versioning dev containers, which lets a development environment evolve together with the application it creates.



Dev containers

You can use this information to understand why and when to choose between available dev containers. You should choose the Starter container when you need basic tasks, or a preconfigured container when your project requires a customized environment (like Node.js or Java) assigned by an administrator.

Starter dev container

The **Starter** dev container provides a basic environment designed for essential tasks like text editing and cloning Git repositories. It includes the **DevOps Code** extension, which offers commands for seamless integration with tools like DevOps Plan on page 217.

This container is available to all users, regardless of teamspace membership. After logging into Code, you will find **Starter** under the **Other Dev Containers** tab. Once launched, it moves to the **Running Dev Containers** tab.

The following video demonstrates two scenarios: one for users without a teamspace, and one for users within a teamspace.

../videos/starter-userspace-not-userspace.mp4

To launch the Starter container, go to the Code landing page and click the **Let's Go** button on the **Starter** container tile. Alternatively, to copy a direct link to the container, click the copy link button on the tile. You can then paste the link into a web browser address bar to open the container.

Preconfigured dev container

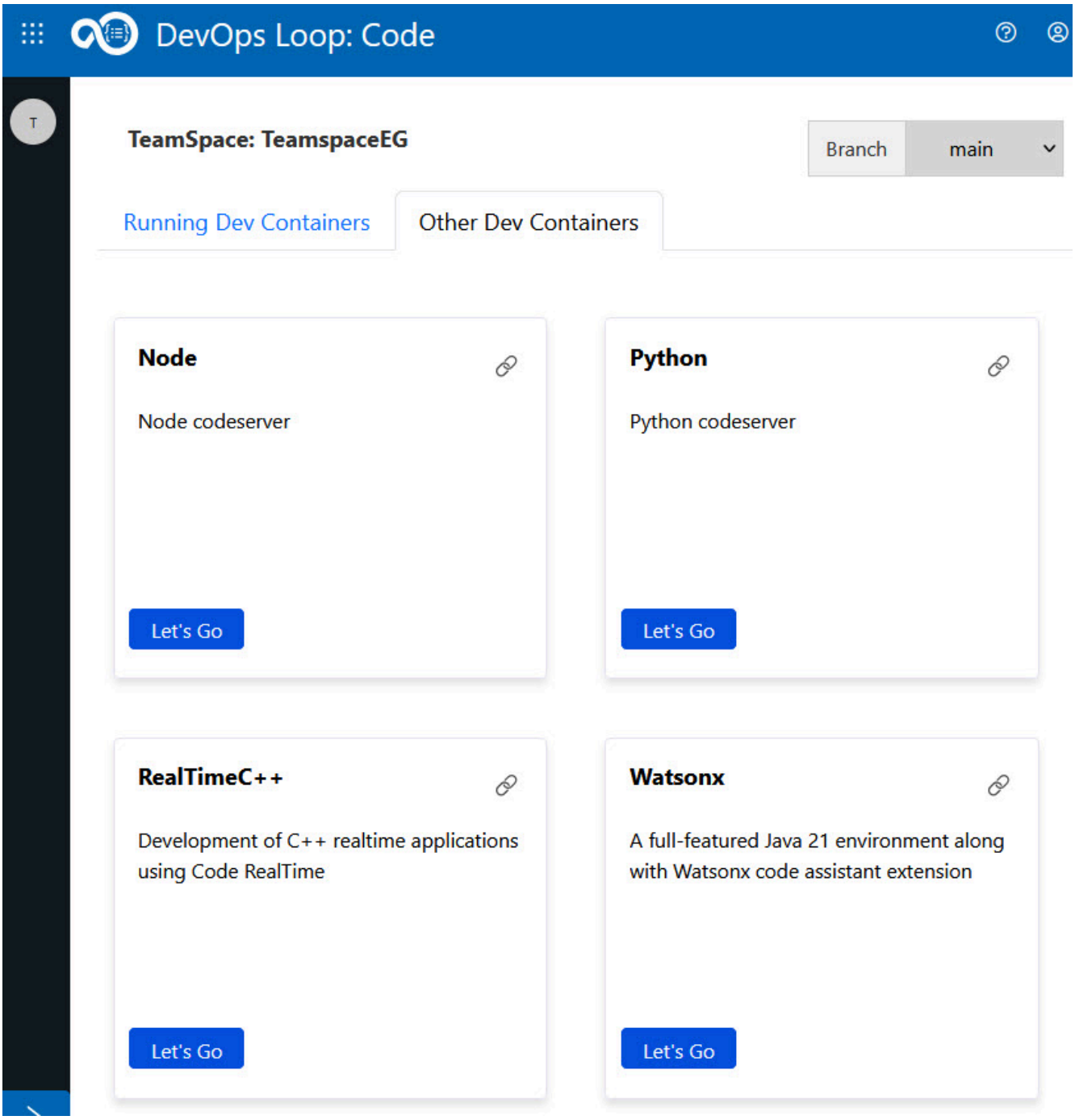
A preconfigured dev container provides a customized development environment tailored to specific project requirements, such as C++, Java, Node, Python, RealTimeC++, Watsonx, etc. A preconfigured dev container is added via a teamspace created and managed by an administrator on page 172.

To access a preconfigured dev container, switch to the teamspace assigned to you by your administrator. Then, select the relevant dev container from the available options. For example, to work in a Node.js environment, click the **Let's Go** button on the Node dev container tile.

Note: If no teamspace appears, it likely means you are not added to one yet. Contact your team admin to request access.

For example only: The image below shows how dev containers appear after switching to an admin-assigned teamspace, such as TeamspaceEG. In this case, you can see preconfigured dev containers alongside the Starter container.

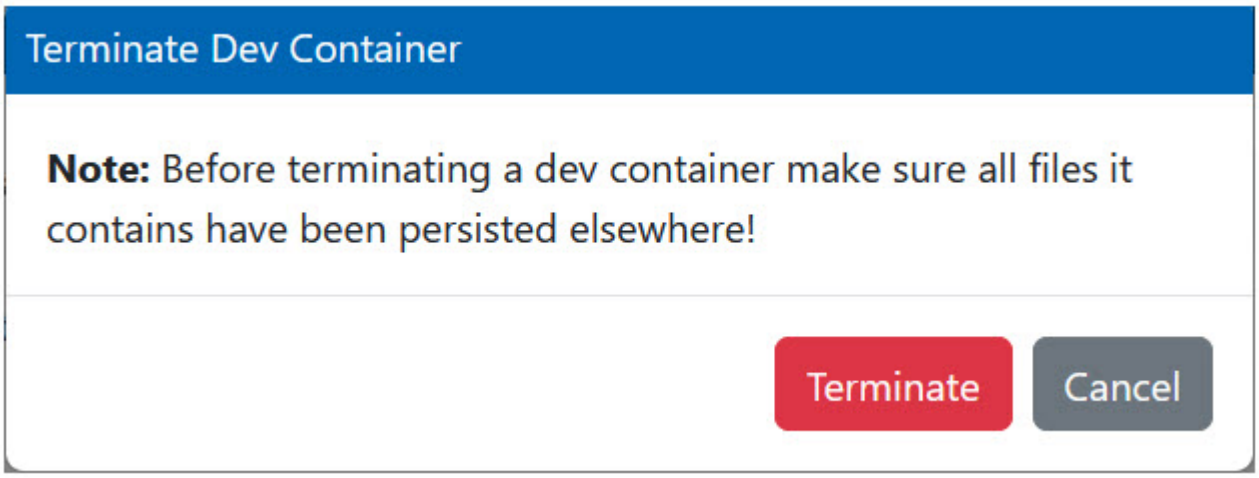
This setup is just an example. The actual teamspace name and available containers might vary.



Note: Administrators can define standard dev containers to ensure consistency across team environments. For more information on configuring dev containers, see the Administrator Guide on page 172.

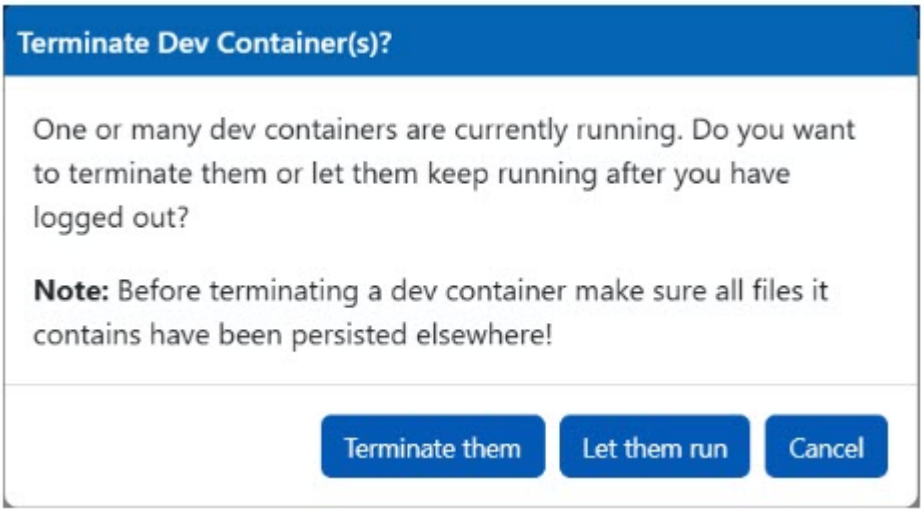
Terminate a dev container

When a dev container starts running, you can see a **Terminate button** on the dev container tile. This action opens the **Terminate Dev Container** dialog.



Important: When you terminate a dev container, all files and unsaved work are deleted. Therefore, you must push your changes to a remote source control repository, such as a Git repository, before termination.

You can also delete all the running dev containers in a namespace by clicking the Logout button, which invokes the **Terminate Dev Container(s)** dialog. When all the dev containers are terminated, the associated user namespace is also deleted.



Viewing logs

Use the Log Viewer in Code when you need to view, filter, or download log output from your dev containers or from Code itself. You can diagnose launch failures, unexpected container behavior, or errors that you want to share with a support team.

Before you begin

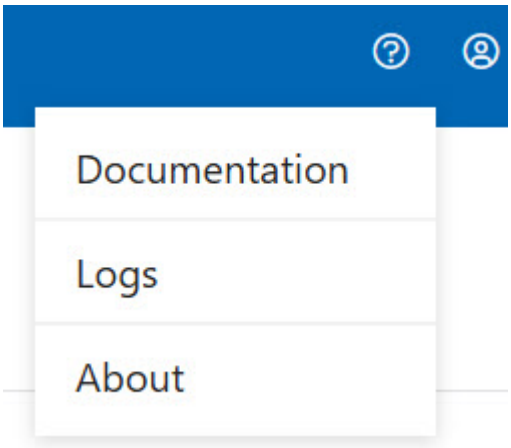
You must have completed the following tasks:

- You have opened the Code landing page.
- You are a member of at least one teamspace, if you want to view the container-level logs.

About this task

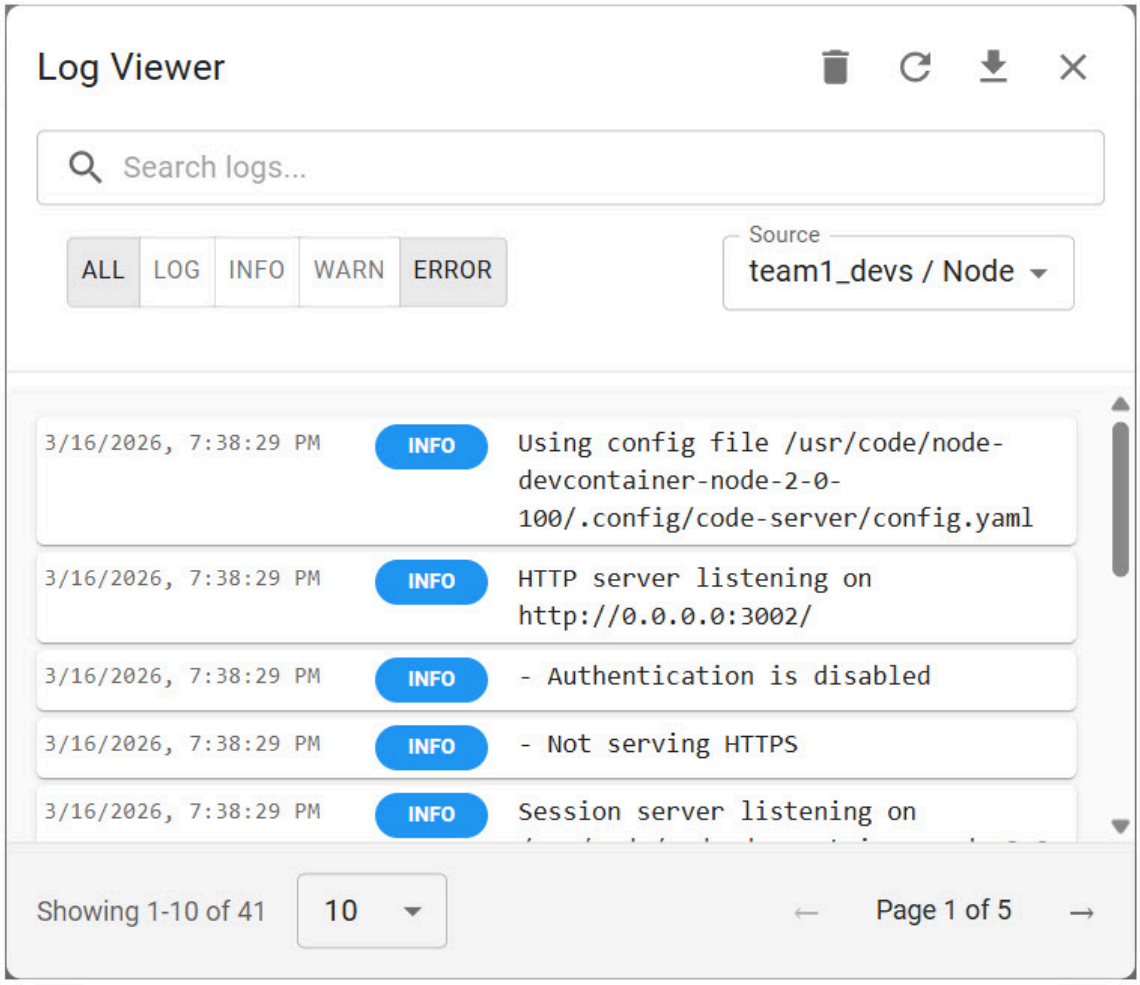
Log Viewer opens as a panel within the Code interface. You access it from the help menu in the top-right corner of the page. Once open, you can switch between Code logs and dev container logs using the **Source** dropdown, apply severity filters, search log entries by keyword, and download captured logs as a file. The Log Viewer captures **trace logs** from your running dev containers and from Code itself. It does not capture output from applications that run inside the dev containers, for example, the output of a Node.js application.

1. Open the Log Viewer:
    a. Onthe Code landing page, click the help icon (top-right).

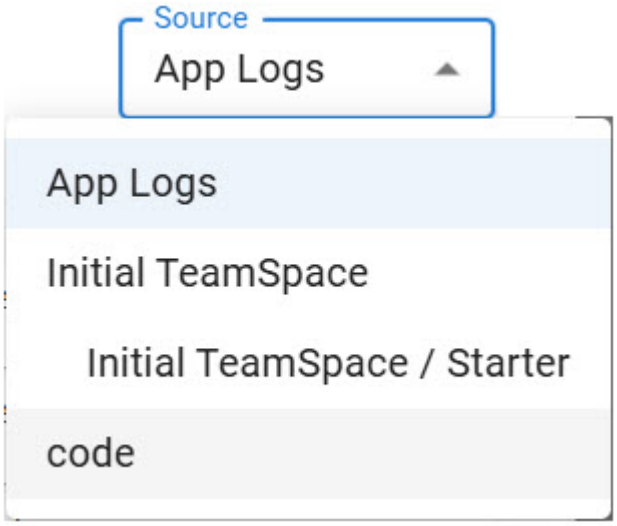


    b. In the dropdown menu, click **Logs**. The Log Viewer panel opens.
2. Select a **log source**:

a. In the upper-right area of the Log Viewer panel, click the **Source** dropdown.



b. To view logs for a container or containers grouped by their teamspace, select the teamspace name from the **Source** dropdown, then select an individual container if needed.



App Logs show internal logs of Code, not logs from apps inside the dev containers.

3. Filter the **log output**:
   a. In the search bar at the top of the panel where there is the **Search logs...** placeholder, type a keyword to narrow entries by matching text.
   b. Click one or more filter buttons to show only entries of that severity level. The following filters are available:

Button	Description
ALL	Displays all log entries regardless of level.
LOG	Shows log entries without severity levels by using a generic format to record plain-text, non-critical data for simple auditing.
INFO	Shows informational messages about normal operational activity.
WARN	Shows warning-level entries, indicating conditions that might lead to errors.
ERROR	Shows error-level log entries, indicating failures that require attention.

Note: You can enable multiple filters simultaneously. For example, selecting both **WARN** and **ERROR** shows only warning and error entries together.

   c. To load the latest entries, click the **Refresh** icon in the top-right area of the panel.
4. View **log entry details**:

a. Locate the log entry you want to inspect. Each row shows a timestamp, a color-coded level badge (for example, **INFO** or **ERROR**), and a short message.
b. Click the expand arrow on the right side of the row. When expanded, the entry displays sections such as Details, which indicates that the entry is expanded and contains further information, and Context, which provides the namespace and pod name associated with the error.
c. Click the collapse arrow to close the expanded view and return to the compact list.

5. Adjust the number of **displayed entries**:
   a. At the bottom of the panel, click the count selector on the left (default is **10**) and select a different value to change how many entries are shown per page.
   b. Use the arrow controls to move between pages.
6. Download the **logs**:
   a. Click the **Download** icon in the top-right area of the Log Viewer panel.
   b. Save the exported file to your local system.

> **Note:** The download exports only the entries that are currently visible based on your active source and filter selections. Adjust your filters before downloading if you need a specific subset of entries.

Results

You have opened the Log Viewer, selected a log source, applied filters, and reviewed or downloaded log entries from your Code environment. You can now use the log output to identify the cause of issues in your dev containers or share the file with a support team.

What to do next

After you review the log output, consider the following next steps:

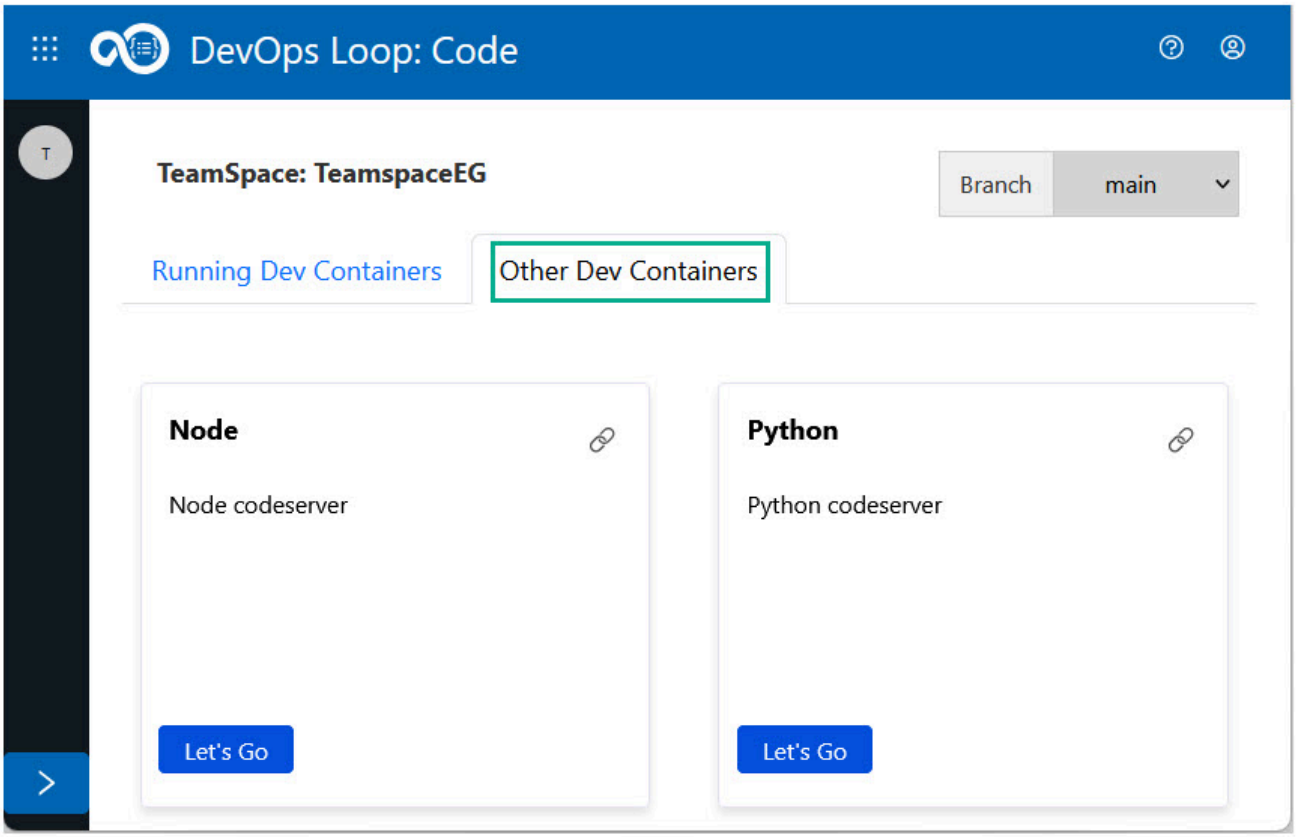
- If you identified an error in the logs, see Configuring dev containers on page 172 to adjust the settings that may be causing the issue.
- If the issue persists, share the downloaded log file with your administrator for further investigation.

Working with dev containers

You can launch a dev container to load your specific development environment and begin active development tasks.

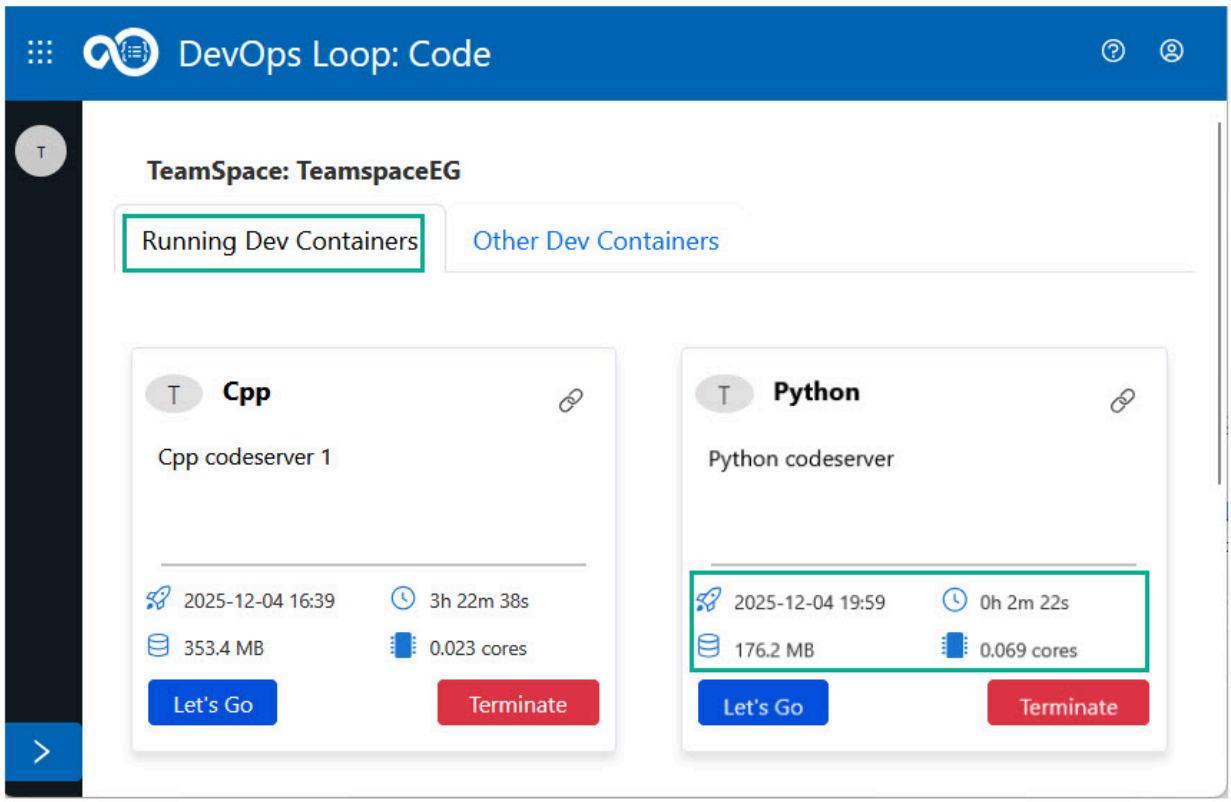
About this task

When you are added to a teamspace, new preconfigured dev containers appear on your landing page under the **Other Dev Containers** tab. Since you have just accessed the workspace for the first time, the **Running Dev Containers** tab will initially be empty. Launching a container moves it to the **Running Dev Containers** tab, where all active environments are managed.



To start working with a dev container:

1. Click the **Let's Go** button on a dev container.
2. The dev container will launch and begin running - this may take some time initially.
3. Once running, the container will move to the **Running Dev Containers** tab.



To optimize performance, Code provides real-time visibility into the resource metrics of your dev containers, such as:

- Launch time and date: The specific date and time the container was started.
- Duration: The total uptime of the container since launch.
- Memory consumption: The real-time RAM usage, measured in Megabytes (MB).
- CPU cores: The number of CPU cores currently utilized by the container.

4. Go to **File** > **Open Folder** to get started, after the Code IDE on page 180 loads.
5. Click **Yes** when prompted with "**Do you trust the authors of the files in this folder?**".
6. Navigate to the workspace directory.
7. You can now start working in your workspace. For example, you can clone a Git repository to populate your workspace with the source code you want to work with.
   Note: This is just a quickstart example. As the environment runs on a VS Code Server, you can install extensions, debug applications with breakpoints and live logs, use terminals and Git integration, and fully customize your dev container with the dependencies you need. Use this as a starting point to explore the full development workflow.

   A video version of the above steps is also available for reference. It demonstrates the full workflow - from accessing Code to running your application in the browser IDE. Watching the video can help you follow along more easily. Learn more about dev containers here on page 162.

   ../videos/new-qs.mp4

Configuring dev containers

As a teamspace admin in Code, you can configure the dev containers so that they are available to the team members.

Before you begin

You must have performed the following tasks:

- Installed Docker and ensured that it is running on your machine.
- Installed Node.js and npm (for Dev Container CLI).
- Created a DockerHub account and enabled it for pushing images.
- Gained access to Control with permission to create repositories.

About this task

You can set up and preconfigure dev containers for users in a teamspace while using Code. You can create and configure the container environment, build an image, publish it to a registry, and make it accessible within the teamspace's configuration repository. These configurations follow the Dev Containers specification and are versioned by pushing them to a Control Git repository.

1. Install the **Dev Container CLI** tool:
   a. Clone the GitHub repository by running the following command:
      ```
      git clone https://github.com/devcontainers/cli.git
      ```
   b. Navigate to the cloned directory and install the following Dev Container CLI:
      ```
      npm install -g @devcontainers/cli
      ```
2. Create the **devcontainer.json** file:
   You can create the devcontainer.json file by defining the metadata such as name, description, build or image source, and runtime Visual Studio Code extensions.
   Here's a **sample devcontainer.json**:
   ```
   {
     "name": "Go Dev Container",
     "description": "A preconfigured Go development environment with code-server support and Go tooling.",
     "build": {
       "dockerfile": "Dockerfile"
     },
     "customizations": {
       "vscode": {
         "extensions": [
           "secure-dev-ops.devops-code",
           "golang.go"
         ]
       }
     }
   }
   ```

Note: In the example above, extensions are specified by their IDs, and will then be downloaded from the Open VSX registry (that is, <publisher.name> format. However, administrators may need to pre-bundle .vsix extension files directly within the dev container image. In such scenarios, especially within

offline environments, you can specify the local .vsix files using a relative path. See Loading extensions from local .vsix files on page 186.

You can also download a sample file here: devcontainer.json

Refer to the Dev Container metadata reference guide to include JSON metadata related to general settings, Docker or Docker Compose-specific options, and tool-specific properties for configuring a devcontainer.js file.

3. Create the **Dockerfile** for custom environment setup:

   You can add a Dockerfile to define the environment and behavior of your development container by specifying the base image, installing tools, setting environment variables, and configuring the container runtime.

   Example **Dockerfile**:

```
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && \
    apt-get install -y \
        sudo \
        curl \
        git \
        ca-certificates \
        build-essential && \
    apt-get clean

# Install Go
ENV GO_VERSION=1.22.3
RUN curl -fsSL https://go.dev/dl/go${GO_VERSION}.linux-amd64.tar.gz | tar -C /usr/local -xz
ENV PATH="/usr/local/go/bin:${PATH}"
ENV GOPATH="/home/1001/go"
ENV PATH="${GOPATH}/bin:${PATH}"

# Install code-server
RUN curl -fsSL https://code-server.dev/install.sh | sh -s -- --version=4.98.0

# Safe for Ubuntu + RHEL images
RUN chmod -R ug+w /etc/pki/ca-trust/source /etc/pki/ca-trust/extracted || true

RUN useradd -m -u 1001 devuser
USER 1001
WORKDIR /home/1001
CMD ["tail", "-f", "/dev/null"]
```

   Key Commands required for the Dockerfile:

   - FROM: Set the base image.
   - ENV: Set the environment variables.
   - RUN: Install dependencies and clean cache to keep the image lean.
   - USER: Run the container as a non-root user (1001) for security.
   - CMD: Keep the container running by tailing a null file.

To create or customize a Dockerfile for your development container, refer to the official Dockerfile reference. This guide provides detailed explanations and examples of all supported commands. You can also explore dev container documentation for guidance on integrating your Dockerfile into a .devcontainer setup.

Note: If DevOps Loop runs using self-signed certificates or private Certificate Authorities (CAs), the admin must add the line RUN chmod -R ug+w /etc/pki/ca-trust/source /etc/pki/ca-trust/extracted in the Dockerfile before the line USER 1001.

You can download a sample Dockerfile here: Dockerfile

4. Build the **dev container** image:
   Navigate to the root **.devcontainer** directory and run the following command to build the dev container image.

```
devcontainer build --workspace-folder . --image-name sarasonia/go-container:1.0
```

   This command reads the devcontainer.json file and executes the Dockerfile. The syntax should be in the following format:

```
devcontainer build --workspace-folder . --image-name <username>/<container-name>:<tag>
```

   where,
   - **<username>**: Docker Hub username
   - **<container-name>**: your container (repository) name
   - **<tag>**: version (for example, 1.0)
5. Update the **devcontainer.json** file to use the image key:

```
{
  "name": "Go Dev Container",
  "description": "A preconfigured Go development environment with code-server support and Go tooling.",
  "build": {
    "dockerfile": "Dockerfile"
  },
  "image": "sarasonia/go-container:1.0",
  "customizations": {
    "vscode": {
      "extensions": [
        "secure-dev-ops.devops-code",
        "golang.go"
      ]
    }
  }
}
```

6. Push the image to **Docker Hub** registry:
   Login and push.

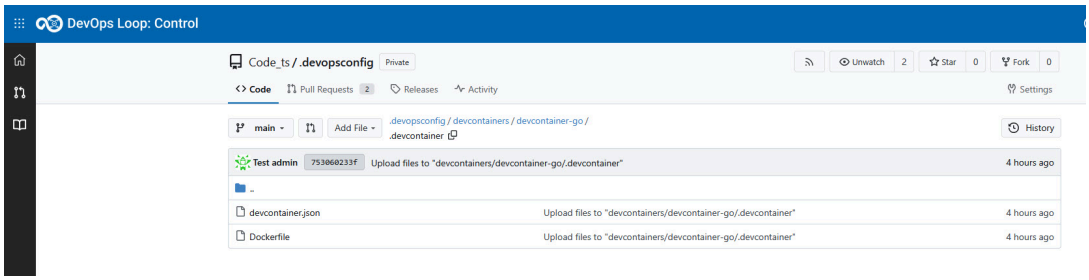
```
docker login
docker push sarasonia/go-container:1.0
```

   After it is pushed, it will be available at: docker.io/devuser/go-container.
7. Access **Control**:

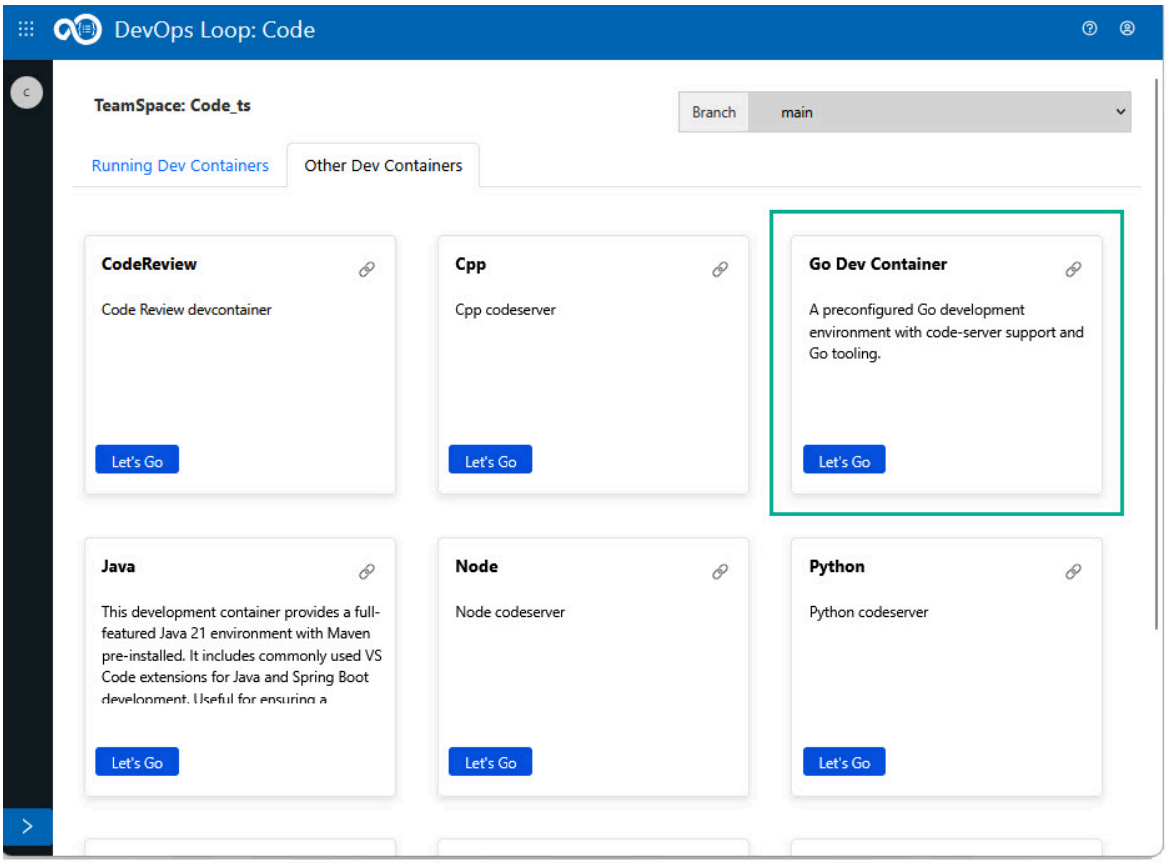
a. Sign in to HCL DevOps Loop. Navigate to your teamspace.
b. Go to the **Control** page. You'll see a repository named .devopsconfig under your teamspace (e.g., TeamspaceEG/.devopsconfig).

8. Upload **devcontainer.json** and **Dockerfile** using Control:
    a. Inside the **.devcontainer** folder of the repo, upload devcontainer.json and Dockerfile.
    b. Commit and push the changes to the main branch.

> **Note:** You can also commit the changes to a different branch other than the main branch. In such a case, you will see different branches on the Branch menu.



9. Start the sample **Go container**:
    a. Go to the Code landing page.
    b. Click the **teamspace** button and switch to your teamspace on page 161. Under **Other Dev Containers**, you will see a new dev container:

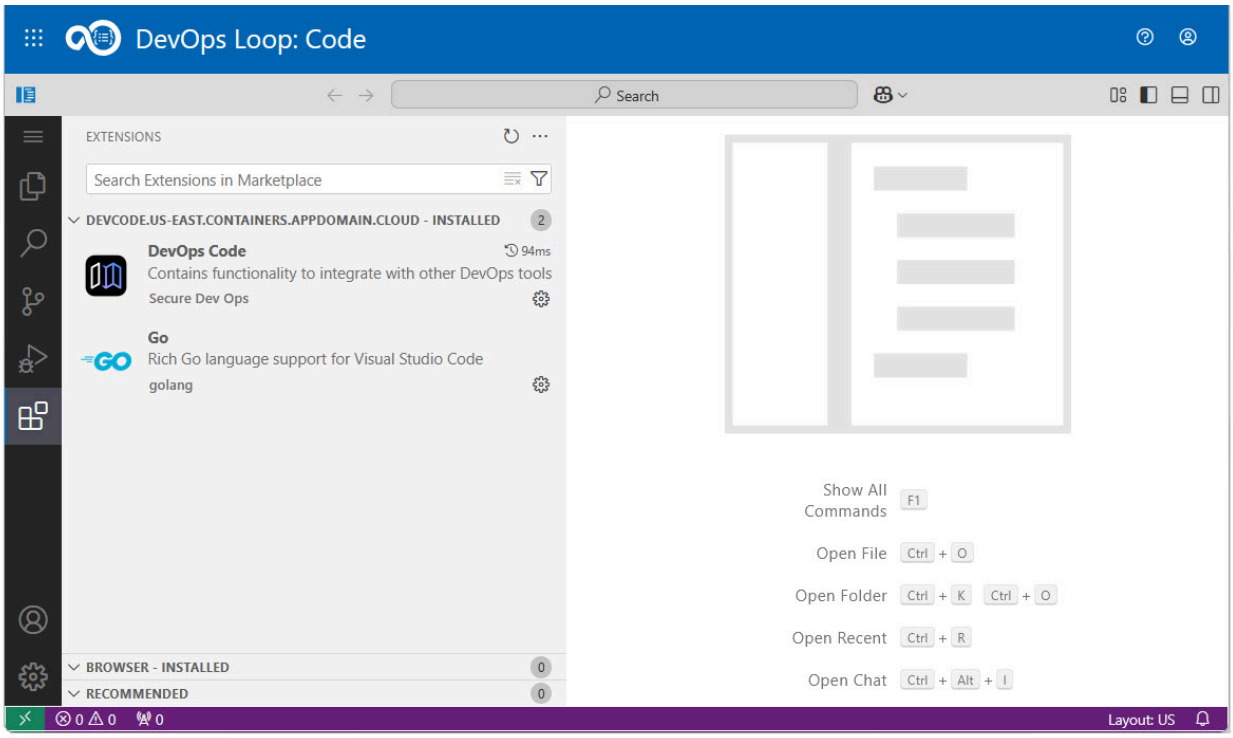


c. Click **Let's Go** to launch the container.

Note: The first-time launch may take a few minutes.

10. View **Visual Studio Code** Extensions:

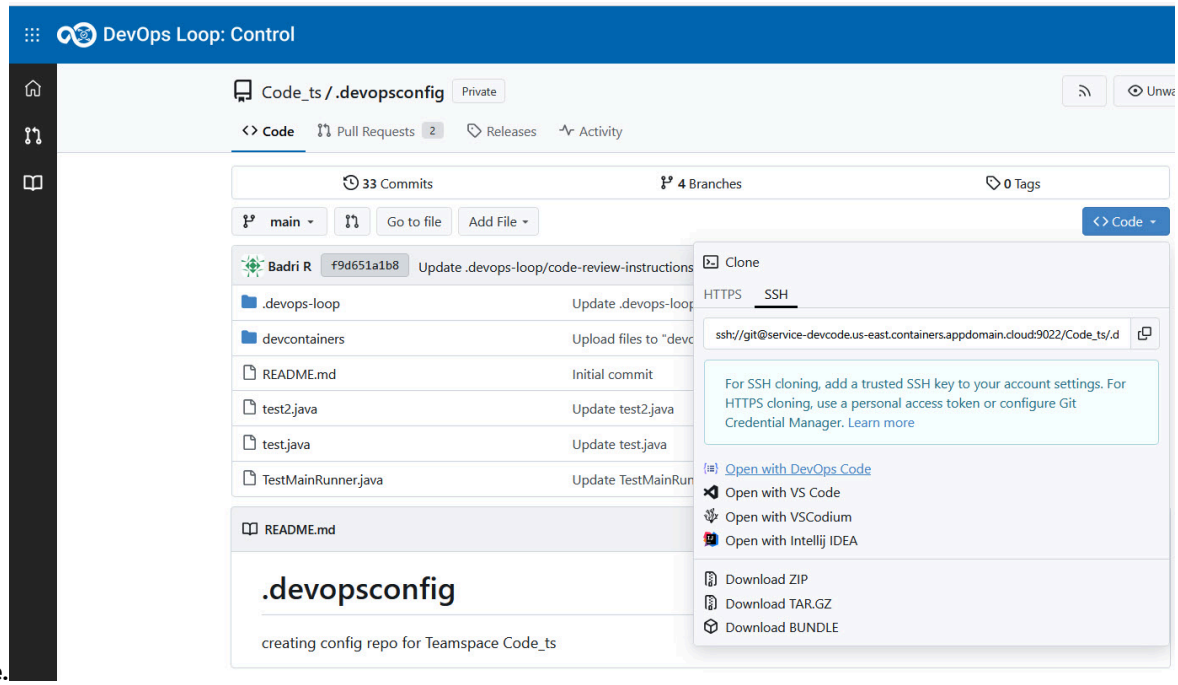
    In the Code IDE instance, click the **Extensions** button to see and verify the extensions you had configured in **devcontainer.json** file.



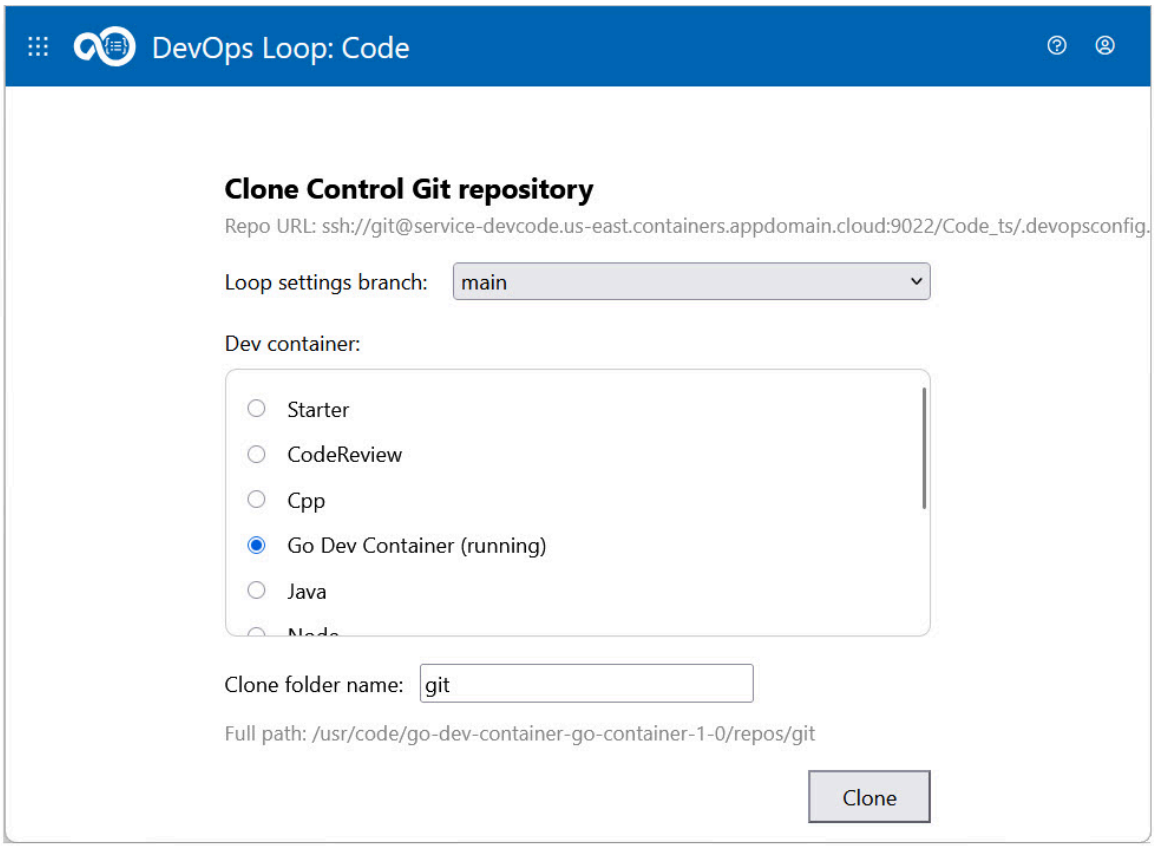
11. Cloning .devopsconfig repo using **DevOps Code**:
    After navigating to the Control page and locating the .devopsconfig repository, you can update your configuration files using the recommended workflow.

    This method is recommended for making multiple changes, as it provides an easier user experience by leveraging the Git functionality within Code.

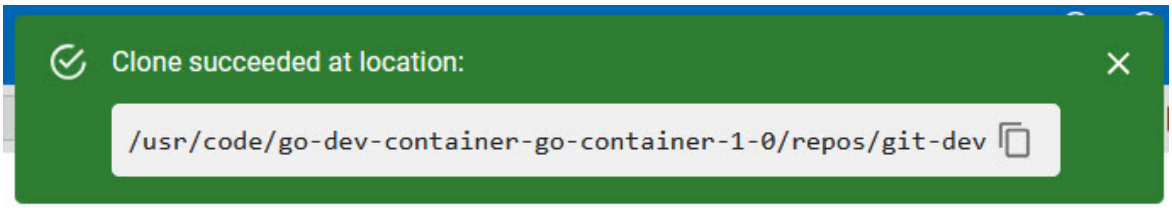
a. Click the **Code** dropdown button next to the repository name, and select **Open with DevOps Code.**



b. This directs to the **Clone Control git repository** page. Here, add the **Loop settings branch**, **Dev container** and a suitable **folder name** for cloning. You can see how the full path changes according to the name you updated in the **Clone folder name** textbox.



c. Click **Clone** to clone the repository. This directs to Code IDE instance. After successful loading of the IDE page, you get a **Clone succeeded at location** message popup. Copy the location.



d. In the Code IDE instance, go to the hamburger menu, select **File** and click **Open Folder**, and paste the location path in the **Go to file** dropdown textbox, and click the **OK** button.
e. Navigate to the devcontainer folder in the left Explorer pane, add/update your devcontainer.json and Dockerfile files, and use the built-in Git tools within **Code** to commit and push the changes to the desired branch of **Control**.

Results

The custom container image has been successfully built and published to the Docker registry. The image is now ready to be configured for use within your teamspace.

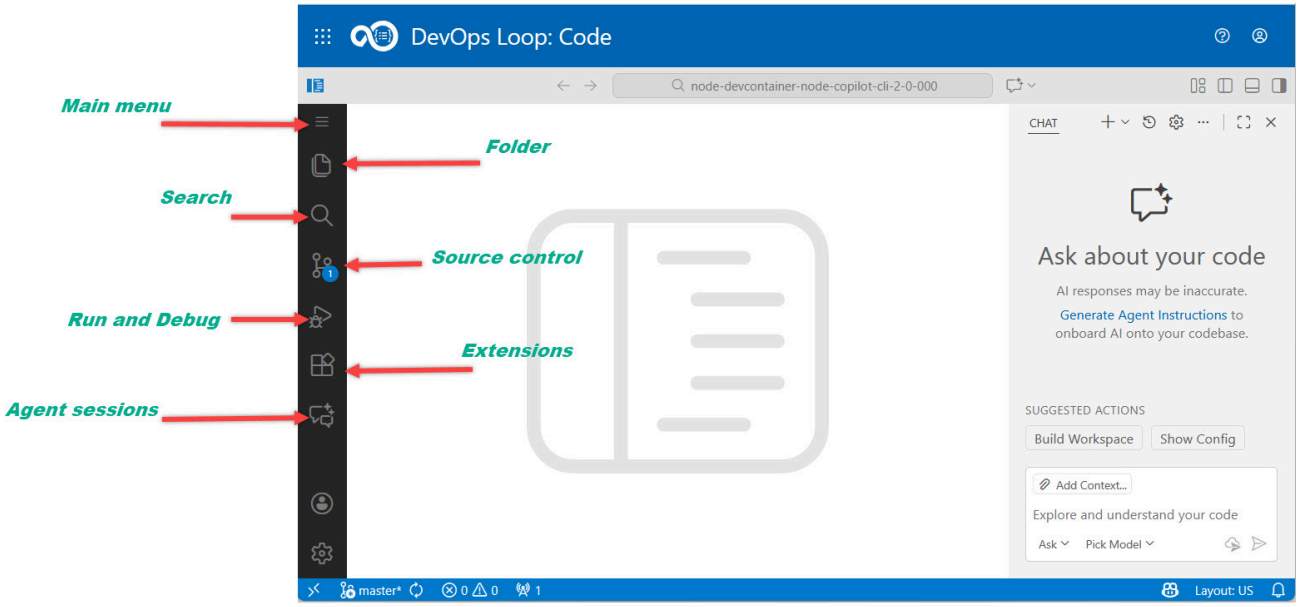
What to do next

- After you configure dev containers for teamspaces, members in a team can launch any dev container directly from their Code landing page.
- If you are working in an air-gapped environment and want to clone repositories correctly, see Fixing Git clone in air-gapped on page 198 environment.

Code IDE user interface

The Code IDE user interface offers key features that are designed to facilitate navigation, file management, source control, debugging, and extension management within the browser IDE.

The following image shows the user interface with a vertical toolbar on the left-hand side of the Code IDE.



The following table lists the main highlights of the Code IDE user interface:

Menu items	Description
Main (hamburger menu)	Opens a collapsible side menu in a dropdown, and includes general menu items such as File, Edit, Selection, View, Go, Terminal, and Help. These menus are mostly the same as those that are present in the main toolbar of Visual Studio Code.
Buttons	
Folder	Opens the Explorer panel that provides tree-like navigation of the workspace structure.
Search	Provides the search and replace functionality for files in the workspace.
Source control	Provides the source control panel from which you can perform commands on Git repositories that are used in your workspace.

Menu items	Description
Run and Debug	Provides common actions for running and debugging applications.
Extensions	Provides a menu to manage extensions.
Agent sessions	Provides a view to manage and monitor ongoing AI-powered autonomous tasks, allowing you to interact with AI agents.

For more information about the UI of the browser IDE, refer to the documentation of Visual Studio Code.

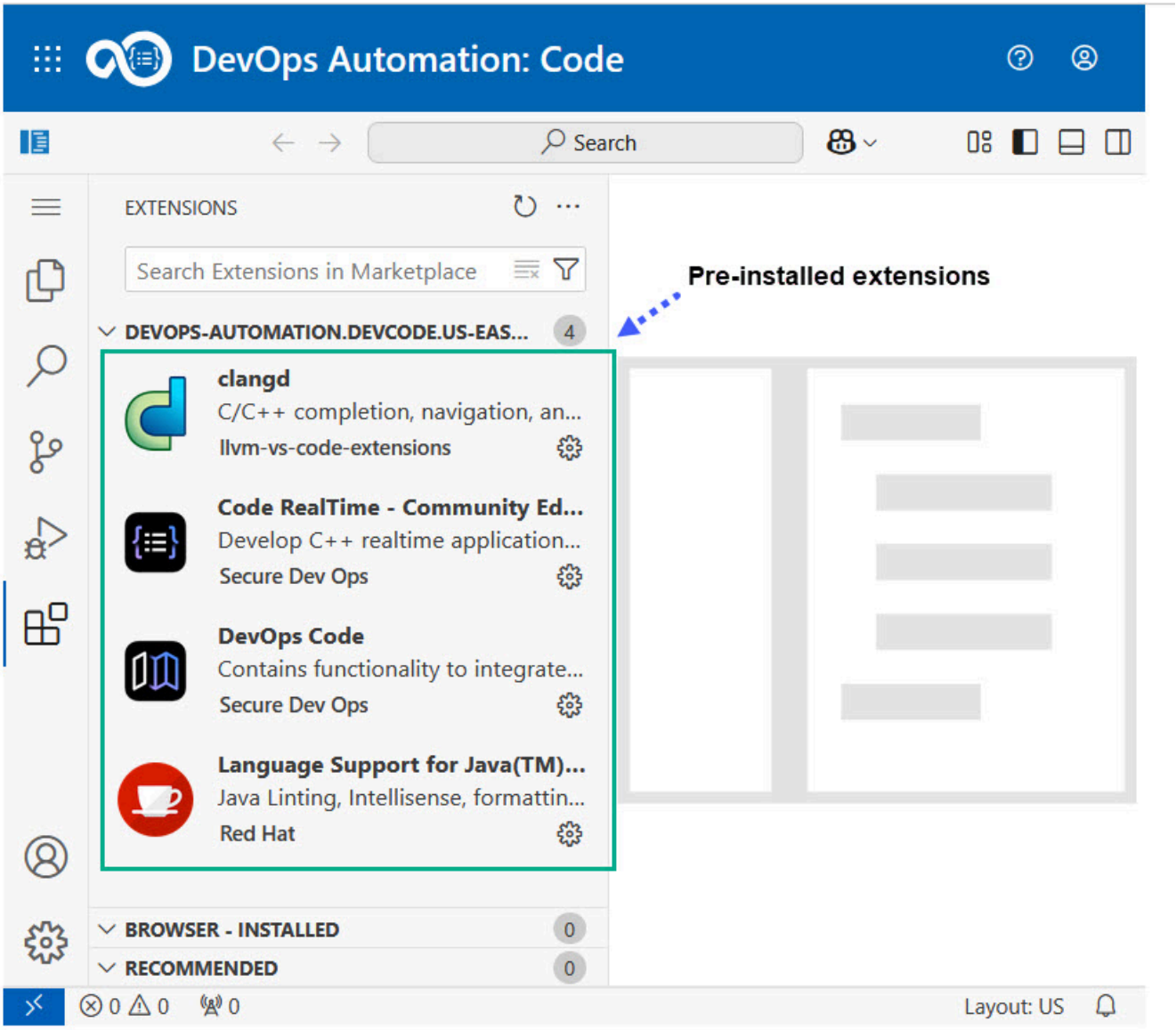
Extensions in dev containers

You can use extensions to specialize the Code IDE, integrating tools for coding, debugging, version control, and more. You can deploy them during initial project setups or to ensure workflow consistency in containerized and offline environments.

Preinstalled extensions

Your administrator can configure dev containers with preinstalled extensions during setup. These extensions are defined in the devcontainer.json file and sourced from the Open VSX Registry. For details on how to configure these in the devcontainer.json, refer to the Administrator Guide on page 172.

The following image shows a simple example of preinstalled extensions available in Code when you launch a dev container.



While working in your browser-based IDE, you can also perform the following tasks:

- Install additional extensions directly from the Open VSX Registry from the Extensions button in the left toolbar of the IDE.
- Install local VSIX files if needed on page 186.
- In an air-gapped environment, install extensions from your organization's configured local extension registry instead of the public Open VSX Registry. For more information, see Configuring extension registry on page 187 and Installing extensions in air-gapped environment on page 190.

File management features

You can transfer files between your local computer and dev container, and simplify your development workflow.

The following section outlines the key features provided by Code for file management:

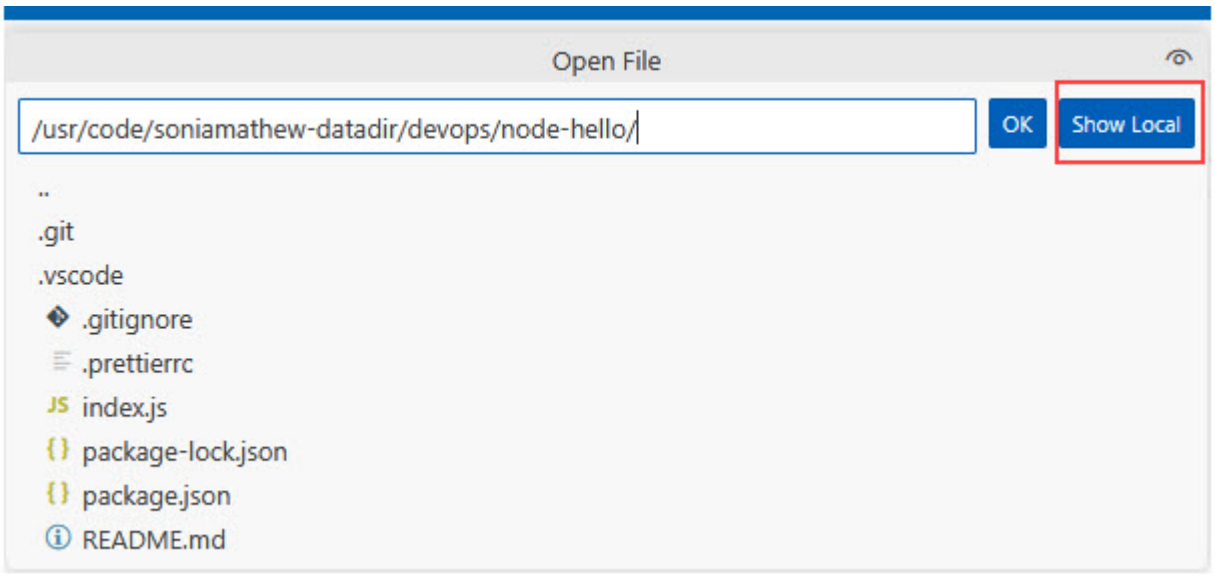
Drag and Drop

Code provides a drag-and-drop feature that simplifies file management, so that you can effortlessly transfer files between your local computer and the dev container. You can simply drag files from your local system and drop them into the Explorer view of the Code IDE to add them to a workspace folder.

../videos/drag-drop.mp4

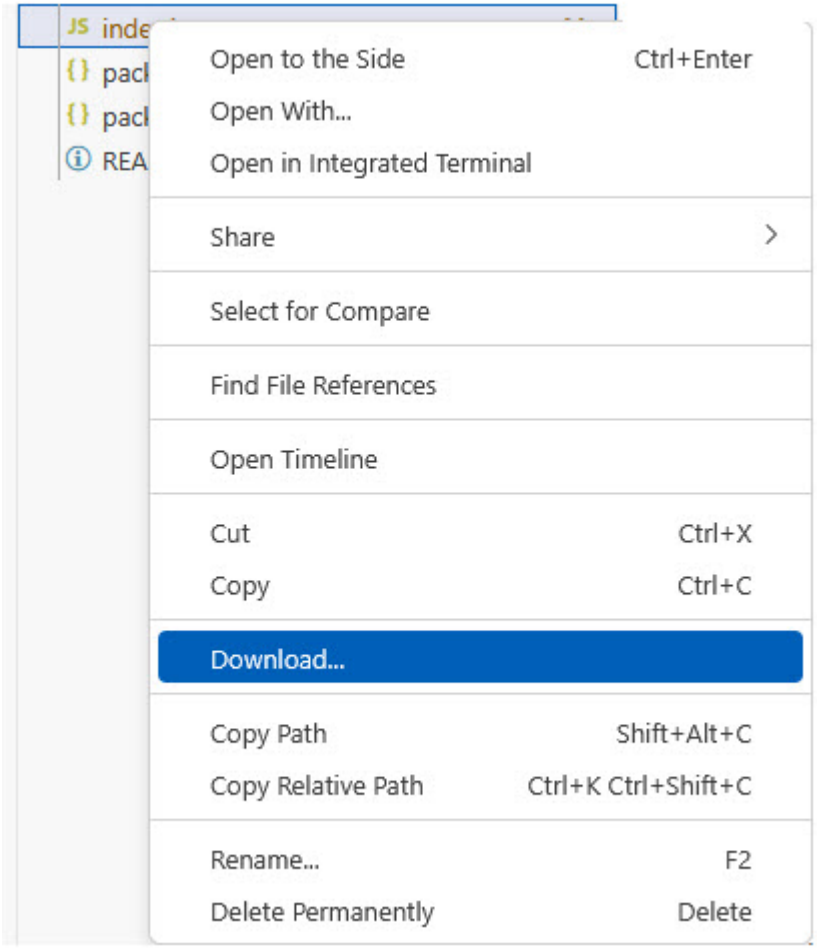
Browse local files

Code provides the **Show Local** button feature, which allows you to browse files from your local machine. This gives you the option to choose between using the file system of your local computer or that of the dev container.



Download

Code provides the **Download** option to easily download files and folders from the server IDE to your local system. Simply right-click on the required file or folder, select the **Download** option, and the file will be transferred to your local machine for backup.



Access applications via automatic port forwarding

You can access applications running within your dev container from your local machine as Code automatically detects and forwards listening ports (7000–8000), to ensure uninterrupted interaction with containerized services.

Note: The dev container uses some ports internally and reserves the port range 7000 - 8000 for your application to use.

When Code detects that an application is running in the dev container and listening on a port, it automatically forwards the port, making it accessible outside the container. For example, you can connect to applications running within the dev container from your local machine.

../videos/auto-port-forwarding.mp4

File system in dev containers

Understanding the file system structure in a dev container helps you organize your work effectively and maintain compatibility across different development environments.

File system tree

You can store your files in or below the /usr/code root directory. This works like a shared directory across all dev containers, ensuring your files remain persistent and accessible.

```
/usr/code/                                    [ROOT DIRECTORY - Shared files
accessible from all dev containers]
│
├── dev-container-1/
│   ├── (your project files)
│   ├── data directory/           [RESERVED - DO NOT MODIFY]
│   └── extension/                [RESERVED - DO NOT MODIFY]
│
├── dev-container-2/
│   ├── (your project files)
│   ├── data directory/           [RESERVED - DO NOT MODIFY]
│   └── extension/                [RESERVED - DO NOT MODIFY]
│
├── dev-container-3/
│   ├── (your project files)
│   ├── data directory/           [RESERVED - DO NOT MODIFY]
│   └── extension/                [RESERVED - DO NOT MODIFY]
│
├── ...
│
└── (additional shared files/repositories)
```

Container-specific directories

You will find each dev container has its own subdirectory under /usr/code where you can create and organize your project files, source code, configurations, and cloned repositories.

You will see two system-managed directories in each container:

- data directory: Contains system-managed data

- extension: Contains system-managed extension files

Note: You must not modify these directories as changes may cause your dev container to malfunction.

Shared files

You can place files directly under /usr/code to make them accessible from any dev container. This is useful for common configurations, shared libraries, or cross-environment projects.

Loading extensions from local `.vsix` files

Using Code, you can install and activate extensions in an offline or air-gapped environment by loading them from local .vsix files specified in the dev container configuration.

About this task

You can load extensions from a local .vsix file by specifying a relative path. This approach allows offline or air-gapped environments to activate extensions and is also necessary for proprietary extensions that should not be published publicly, without requiring access to the Open VSX registry.

1. Place your .vsix extension files into a local folder, such as extensions/ inside a cloned .devopsconfig repository: This allows offline environments to access the extensions without connecting to the Open VSX registry.
2. Reference the local .vsix files in your dev container configuration file (devcontainer.json):

   devcontainer.json example:

   ```
   {
     "name": "Node",
     "description": "Node codeserver environment with offline extensions",
     "build": {
       "dockerfile": "./Dockerfile"
     },
     "customizations": {
       "vscode": {
         "extensions": [
           "IBM.wca-core-1.6.2.vsix",
           "ms-python.python-2025.14.0.vsix"
         ]
       }
     }
   }
   ```

   You can specify the exact version of each extension you want to install and control updates centrally.
3. Update your Dockerfile to copy the extensions folder into the container during build:

   Dockerfile example:

   ```
   FROM mcr.microsoft.com/devcontainers/javascript-node:1-22-bookworm

   ENV DEBIAN_FRONTEND=noninteractive

   RUN apt-get update && \
   ```

```
    apt-get install -y sudo curl git gnupg pass pinentry-tty && \
    apt-get clean

RUN curl -fsSL
 https://github.com/git-ecosystem/git-credential-manager/releases/download/v2.6.1/gcm-linux_amd64.2.6.1.
tar.gz -o gcm.tar.gz \
    && tar -xvf gcm.tar.gz -C /usr/local/bin \
    && rm -rf gcm.tar.gz \
    && git-credential-manager configure

RUN curl -fsSL https://code-server.dev/install.sh | sh -s -- --version=4.103.2

COPY extensions /opt/extensions/

USER 1001

CMD ["tail", "-f", "/dev/null"]
```

This ensures the extensions are installed automatically when the container is set up.

4. Build and run the dev container:
   The extension manager will detect the local .vsix files and install them during container startup.

Results

You have successfully installed and activated extensions in your dev container using local .vsix files. The extensions are now immediately available for use.

What to do next

If you need to manually search for and install additional extensions inside an active air-gapped dev container from your local extension registry, see Installing extensions in air-gapped environment on page 190.

Configuring extension registry

You can configure a local or an on-premises extension registry in Code as an admin to ensure that only approved extensions are available to team members within your teamspace.

Before you begin

Before you begin:

- Gained access to DevOps Control with permission to create repositories.
- Download the required .vsix extension files from a trusted source, or use non-published extension files that you have access to. You can visit the specific extension's page in the Open VSX registry, download the desired stable release's .vsix file. Generally, prerelease versions are also listed in Open VSX and only release versions are recommended for deploying.
- Read Authenticating with DevOps Code UI on page 192.

About this task

An extension registry decides which extensions are available in your teamspace. Regardless of which configuration you choose, only administrators can upload extensions to the registry, and teamspace members can only view and install the extensions that an administrator has approved and published.

There are three ways to configure the registry:

1. **Control repository storage** - Extensions are stored and versioned in the **.devopsconfig Control** repository. Requires a manual commit and push after each deployment.
2. **Persistent Volume Claim (PVC) storage** - Extensions are stored on a persistent volume. Deployment is automated; no commit or push required. The default capacity of the storage is 20 GB.
3. **On-premises Open VSX instance** - Extensions are hosted and managed on your own Open VSX instance.

The following table compares the three configurations:

Table 5. Registry configuration comparison

Configuration	`extensions_gallery` value	`storage_type` value	Commit/push required?
Control repository storage	`local`	`control`	Yes, after every deployment
PVC storage	`local`	`pvc`	No, deployment is automated
On-premises Open VSX	`<your-onprem-openvsx-url>/vscode/gallery/extensionquery`	Not applicable	Not applicable

The **DevOps Code** extension provides commands that you can use as an administrator to deploy and manage extensions within the extension registry. The key commands provided by the **DevOps Code** extension in the cloud IDE are:

- **DevOps Code: Deploy to Extension Registry** - Deploys extension files to the registry.
- **DevOps Code: Fetch from Extension Registry** - Refreshes the registry after new extensions are pushed to the repository.

Note: These commands apply only when `extensions_gallery` is set to `local` (Control repository or PVC storage). They are not applicable when using an on-premises Open VSX instance.

Set Registry Configuration

1. Choose one of the three configurations and update `settings.json` in the repository root accordingly:

   Based on the comparison above, decide whether you want to use Control repository storage, PVC storage, or an on-premises Open VSX instance, then add the corresponding values to `settings.json` as shown below.

   a. Option A - Control repository storage:

   ```
   {
       "extensions_gallery": "local",
   ```

```
    "storage_type": "control"
}
```

b. Option B - Persistent Volume Claim (PVC) storage:

```
{
    "extensions_gallery": "local",
    "storage_type": "pvc"
}
```

c. Option C - On-premises Open VSX instance:

Set `extensions_gallery` to the hosted URL of your Open VSX instance, appended with the `/vscode/gallery/extensionquery` path. The `storage_type` parameter is not applicable for this configuration, since extensions are hosted and managed by your on-premises Open VSX instance rather than by Code.

```
{
    "extensions_gallery": "<your-onprem-openvsx-url>/vscode/gallery/extensionquery"
}
```

d. Supported parameter values:

The `extensions_gallery` parameter supports these values:

- `local`: Uses the administrator-controlled registry (Control repository or PVC storage).
- `disabled`: Prevents the user from installing new extensions from the Code IDE.
- `default`: Uses the public Open VSX registry.
- A URL ending in `/vscode/gallery/extensionquery`: Uses an on-premises Open VSX instance hosted at that URL.

The `storage_type` parameter supports these values:

- `control`: Stores extensions in the .devopsconfig **Control** repository.
- `pvc`: Stores extensions on a persistent volume.

2. Add approved extensions in Code:

Note: This step applies only to local registry configurations (Control repository or PVC storage). It is not applicable when using an on-premises Open VSX instance.

a. Select **Open with DevOps Code** to clone the .devopsconfig Control repository in on page 192 Code.
b. After cloning the main branch of the .devopsconfig repository in Code, create an `extensions` folder in the repository root if one does not exist. Note that you can also create other branches and version the extensions on different branches.
c. Upload the `.vsix` files into the `extensions` folder.

3. Deploy and publish the extensions:

Note: This step applies only to local registry configurations (Control repository or PVC storage). It is not applicable when using an on-premises Open VSX instance.

a. Right-click the `.vsix` file in the Explorer pane and select **Deploy to Extension Registry**.
A dialog confirms that the extension was deployed successfully. If you chose Control repository storage, the extension folder is created inside `extensions` in the Control repository, and the extension is not available to teamspace members until you commit and push the changes. If you chose PVC storage, deployment is automated and the extension becomes available without any further commit or push.

4. If you chose Control repository storage, commit and push the changes to **Control**. Skip this step if you chose PVC storage, where deployment is automated:

a. Commit the following items:
- `extensions` folder
- `settings.json` file

b. Push these changes to the Control repository.
After the push completes, the extension is published to the registry and is available to all team members.

5. Refresh the extension registry:

The registry does not automatically detect new extensions immediately after a push.

a. As an administrator, open the Command Palette and run **DevOps Code: Fetch from Extension Registry** to refresh the registry content and test that the deployment succeeded.
A notification confirms that the registry refresh is complete, and you can search and see the new extension(s). Other teamspace members do not need to run this command; searching for the extension in the Extensions view automatically triggers a rescan and shows the newly deployed extension in the results.

Results

The extensions are now available in the registry. Members in a teamspace can view and install only the extensions that you have added and published.

What to do next

- Install the extensions approved by administrator from the Extensions view in Code.
- Learn how to install extensions manually for air-gapped environment. See Installing extensions in air-gapped environment on page 190.

Installing extensions in air-gapped environment

You can manually install required extensions inside a custom dev container when operating in an air-gapped environment.

Before you begin

Before installing extensions in an air-gapped dev container, review the following prerequisites:

- **Administrator prerequisite:** Ensure that an administrator has configured your local extension registry and published the required extensions. For details, see Configuring extension registry on page 187.
- Ensure that an administrator has configured the custom dev container specification for your teamspace. For details, see Configuring dev containers on page 172.
- Learn how extension installation and lifecycle management work within containers by reading Extensions in dev containers on page 181.
- Familiarize yourself with the layout of the Code browser IDE by reviewing Code IDE user interface on page 180.
- If you prefer to automate extension installation during container builds rather than installing them manually, see Loading extensions from local .vsix files on page 186.

About this task

In an air-gapped environment, custom dev containers (such as a Java dev container) do not automatically install extensions upon launch — except for the DevOps Code extension.

Once an administrator has configured the local extension registry, developers can manually locate and install approved extensions inside their running dev container session.

1. Launch your custom dev container in Code.
2. In the primary sidebar, click the **Extensions** icon to open the Extensions view.
3. Search for the required extension hosted in your local registry and click **Install**.

Results

The selected extensions are now active and operational inside your active dev container session.

Git authentication for dev containers

You can use Git within a dev container to ensure version control and collaboration while keeping your environment isolated.

There are many authentication methods specific to configuring Git in dev containers, including UI feature, personal access tokens and device code authentication.

- DevOps Code UI (Recommended and Easier) on page 192
- Personal Access Token (PAT) on page 195
- Device Code (GPG) on page 196

If you are working in an air-gapped environment, a workaround is required to clone repositories. See Fixing Git clone in air-gapped environment on page 198.

Note: For information on configuring Git credentials using Control Repositories, see Managing Git authentication for developers.

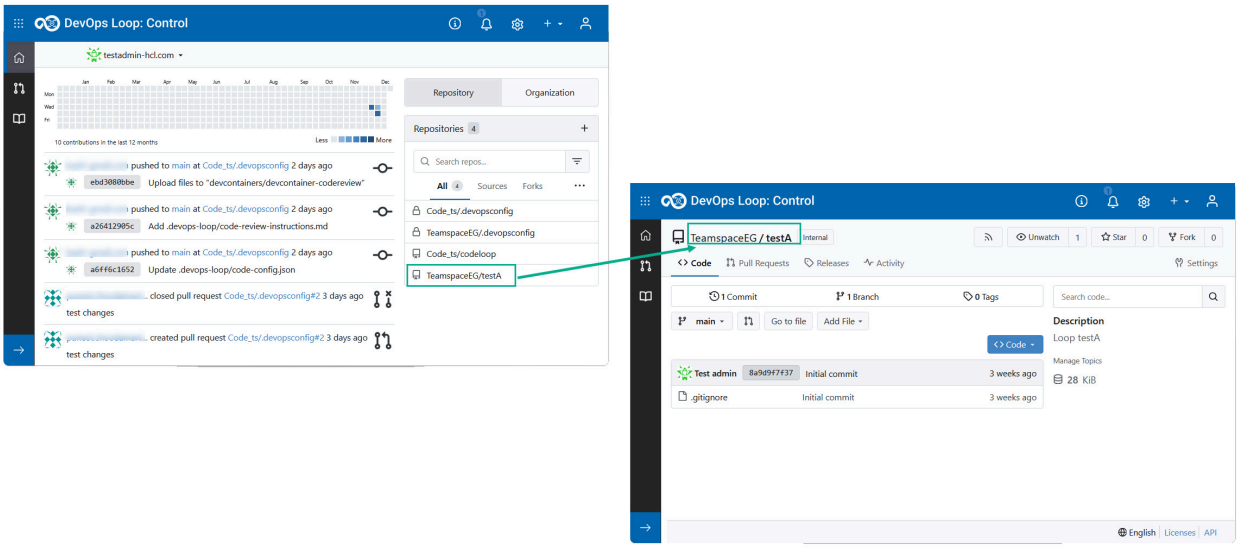
Authenticating with DevOps Code UI (Recommended)

You can create a personal access token in Control, and then use that as your password when authenticating from Code. This type of authentication is based on SSH keys, which are automatically generated for your dev container and registered with Control.

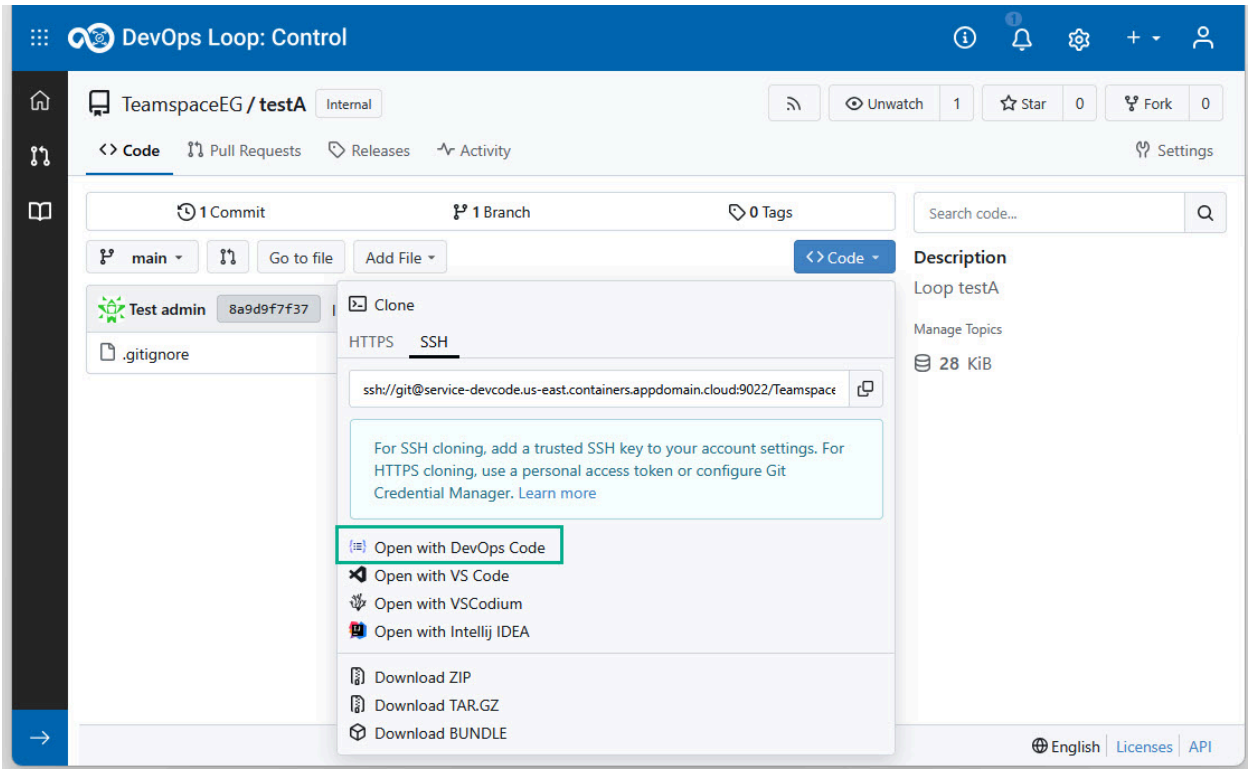
About this task

You can use this procedure to clone a **Control** repository and push changes using **Code**.

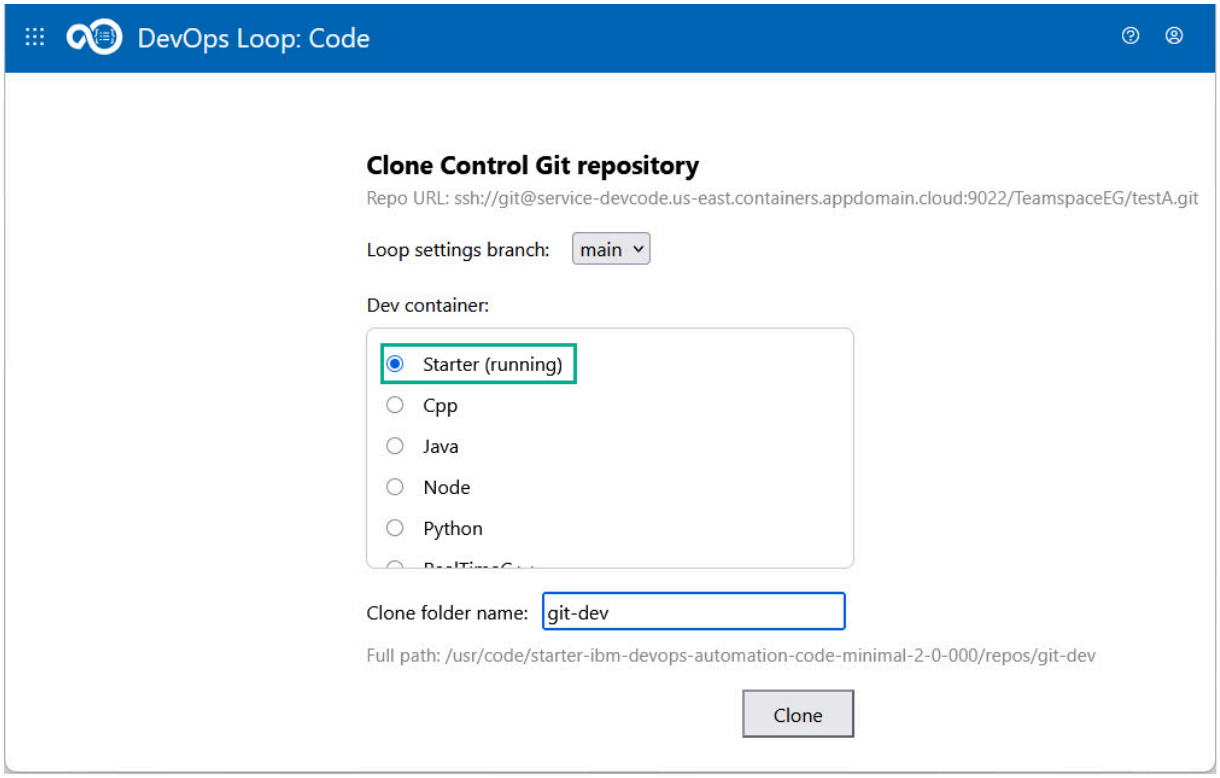
1. Navigate to the Control repository you wish to clone:



2. Click **Open with DevOps Code**.



3. You will be redirected to the **Clone Control Git repository** page.



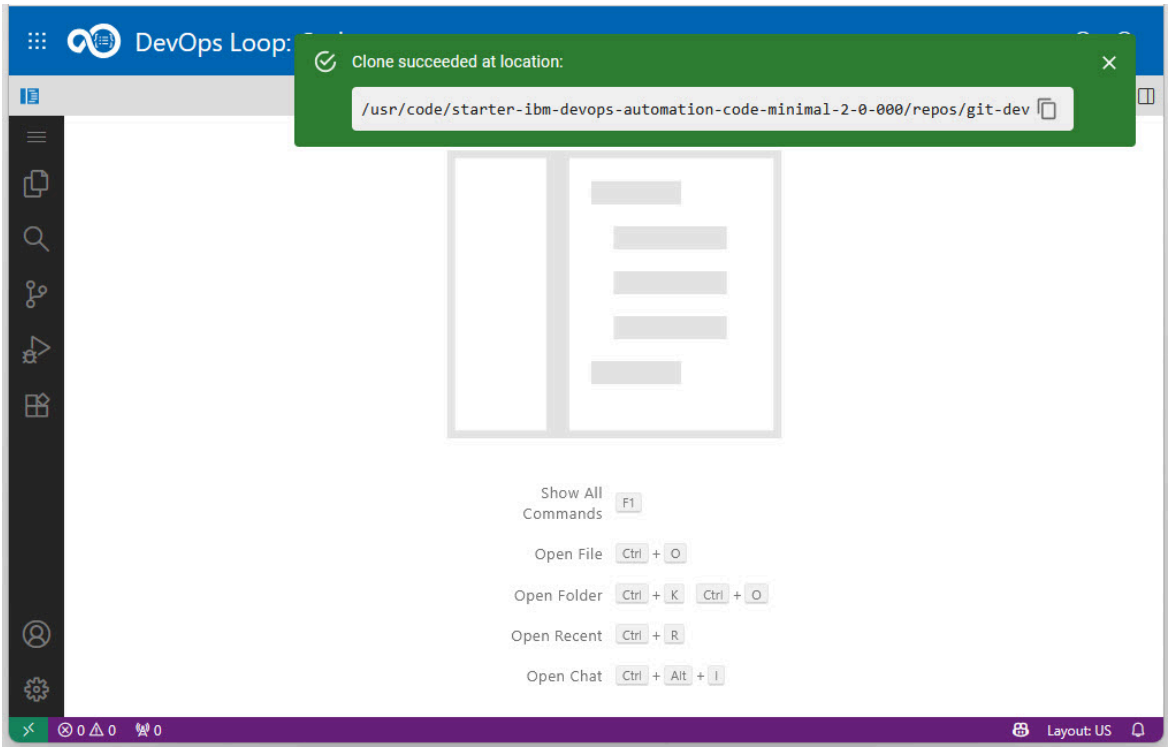
Configure the repository settings:

a. Select the **Loop settings** branch. This is the branch from where you can fetch available dev containers that you choose to use.
b. Select **Dev container**. The procedure will be faster if you select a dev container that is already running. Otherwise, the dev container will first be launched before beginning a cloning.
c. Enter a valid folder name in the **Clone folder name** field.

> **Note:** The **Full path** field will automatically update on page 184 to reflect the folder name you enter.

d. Click **Clone**. The Code IDE instance will open, and the repository will be cloned.

4. Copy the clone location:
   a. Wait for the IDE page to load completely.
   b. Locate the **Clone succeeded at location** notification popup.



   c. Copy the file path displayed in the message.
   d. Open something from the cloned repository in the Code IDE. Choose one of the following alternatives:
      - Open the repository root folder (which you copied above), or one of its subfolders, as a workspace folder: **File > Open Folder**
      - Add the repository root folder (which you copied above), or one of its subfolders, to a workspace that is already open: **File > Add Folder to Workspace**
      - Open a .code-workspace file from the cloned repository: **File > Open Workspace from File**

5. Commit and push your changes:
   Use the built-in Git source control tools within the IDE to commit your updates and push them to the desired branch.

Authenticating Git with personal access token

You can securely clone a Git repository from your Control system. You can use it as a secure alternative to other Git authentication methods for accessing files in your dev container.

Before you begin

You must have performed the following task:

- Read File system in the dev container on page 184.
- Read How to generate and use Personal Access Token in Control.

About this task

The following method outlines how to clone a Git repository in Code using a Personal Access Token (PAT).

1. **Generate a Personal Access Token**
   a. Navigate to your **Settings** in Control to generate a PAT.
   b. To generate a personal access token, click the **Access Tokens** tab in Settings.
   c. Add the Token Name, and set all permissions to Read and Write.
2. **Construct the clone URL for repository owned by username or organization ID**
   a. The URL for cloning is structured with your username and the PAT. For example, assume your username is code@example.com, DevOps Loop is deployed at mydomain.com and your Control repository has the name repo-name, then the cloning URL will look like this:

      https://code-example.com:PAT@mydomain.com/control/code-example.com/repo-name

   b. Note that the @ in the username is replaced with a hyphen (-) in the URL.
   c. Replace PAT with the personal access token you generated in Control.

      Note: When creating a repository in Control, the **Owner** tab gives you two options: username or teamspace/organization ID. This choice dictates the correct structure for the clone URL.

      For a teamspace-owned repository, the teamspace ID must be used as the organization_identifier (the second-to-last part of the URL path) for the clone operation to succeed.

      https://code-example.com:a1b39ac1e546af4dc664ba36f4b12e77f8dad2da@mydomain.com/control/teamspace ID/repo-name

3. **Clone the Repository**
   You can clone the repository either via the terminal or UI:
   Using Terminal:
   a. Run the full git clone command with the constructed URL

      git clone https://code-example.com:<PAT>@mydomain.com/control/code-example.com/repo-name

   Using UI:

a. Go to the Source Control page in the IDE.
b. Click the Clone Repository button in the UI.
c. Paste the constructed clone URL into the input box.
d. Choose the target folder (e.g., /usr/code/repos).
e. Click Clone to start cloning.

Results

The Git repository is successfully cloned to your target folder. You can now begin working with the cloned files in your dev container.

Authenticating Git with device code

You can set up a GPG key and password store within your dev container to perform Git operations securely and establish a robust authentication layer to clone, commit, and push changes to your repository.

Before you begin

You must have performed the following task:

- Read File system in the dev container on page 184.

About this task

The process involves creating a GPG key, initializing a password store, and then using a device code or passphrase to perform Git operations. https://docs.github.com/en/authentication/managing-commit-signature-verification/generating-a-new-gpg-key

1. **Launch the dev container**
   a. From the launcher in DevOps Loop, click the **Code** button.
   b. Run the dev container to start the Code IDE instance.
   c. After the container is running, open a new terminal by clicking the **Explorer** button in the right vertical toolbar and selecting Terminal.

   All subsequent steps must be run as commands from this terminal.
2. **Configure a GPG Key**
   a. In the terminal, run the following command to create a GPG key with a passphrase

      ```
      gpg --full-generate-key
      ```

   b. When prompted, select "RSA and RSA (default)", which is the first option, as the key type.
   c. Specify the desired key size and how long the key should be valid.
   d. Once the key longevity is provided, you will get a prompt showing the key's expiration date. You need to confirm if this date is correct.
   e. The key identification process will then ask for a User ID, which includes your real name, email address, and an optional comment. During this step, you may be instructed to perform actions like moving your mouse or typing on the keyboard to generate enough "entropy".
   f. To protect your new key, you will be prompted to enter a passphrase.

g. After the key is generated, you will see a line that says

   gpg: key <GPG Key> marked as ultimately trusted

   This confirms the key's creation.

h. If the key is not printed properly, use this command:

   gpg --list-secret-keys --keyid-format LONG

3. **Initialize the Password Store**
   - Use the following command to initialize the password store with your key.

     pass init <GPG Key>

     A message similar to the following confirms that the password store has been initialized and a directory (e.g., /usr/code/starter-ibm-devops-automation-code-minimal-1-0-300/.password-store/) has been created:

     mkdir: created directory '/usr/code/starter-ibm-devops-automation-code-minimal-1-0-300/.password-store/'

     Password store initialized for <GPG Key ID>

4. **Export an environment variable**
   You have to export an environment variable each time you open a new terminal.
   export GPG_TTY=$(tty)
5. **Perform Git Operations via Terminal**
   After the setup is complete, you can use a device code to perform Git operations.The device code is only valid as long as the GPG key is valid. If the key expires (for example, in one day), you will need to repeat the entire setup process the next day. When a session expires, you can use the passphrase to perform further Git operations.
   Example: Cloning, adding, committing, and pushing changes to the Control repository
   a. Copy the repository link from Control Repository, and run the following command to clone it:

      git clone <control-repo-link>

      Note: During the clone operation, you are prompted to authenticate by visiting a link provided in the terminal, entering the provided device code, and then granting access to the Git credential manager by clicking Yes.

   b. After cloning, you can add a file, and then commit and push the changes to the repository. This process is similar to working with a standard GitHub repository.

      When you run git push, you are prompted to provide your passphrase to complete the operation.

6. **Alternatively, using the Git UI**
   a. From the left toolbar, click File > Open folder and select the root directory.
   b. Click the Clone Repository button.
   c. Paste the Control repository link when prompted.

d. Authenticate by visiting the provided link and entering the device code, then approve the Git credential manager prompt.
e. Once cloned, use the UI to add, commit, and push changes.
f. When pushing, enter your passphrase when prompted to complete the operation.

Fixing Git clone in air-gapped environment

In air-gapped environments, URLs that start with `service-`, such as the default Control SSH URL, are not accessible from the dev container terminal. You can replace this URL with the internal cluster-local service address to clone a repository.

Before you begin

You must have the following information:

- Access to a dev container terminal running in an air-gapped environment.

About this task

In air-gapped environments, URLs that start with `service-`, for example `git@service-loop.loopairgapvpc1.us-east.containers.appdomain.cloud:30328`, point to an external endpoint that air-gapped virtual private clouds (VPCs) cannot reach. As a result, you cannot use the default Control SSH URL to clone a repository directly from the dev container terminal.

To work around this restriction, replace the external Control SSH hostname with the internal cluster-local service address before you run git clone.

1. **Identify the failing Git command**
   a. Locate the git clone command, or any other Git SSH operation, that references the external Control SSH URL.
   b. An example of a URL that fails in an air-gapped environment is shown below:

      git clone ssh://git@service-loop.loopairgapvpc1.us-east.containers.appdomain.cloud:30328/test_ts/.devopsconfig.git

2. **Replace the Control SSH hostname**
   a. Replace the hostname portion of the URL with the internal cluster-local service address, using the following format, where *platform_namespace* is typically `devops-loop`:

      git@devops-loop-control-ssh.<platform_namespace>.svc.cluster.local:22

   b. Change the port number to (`22`) and keep the repository path unchanged. Only the hostname and port may be modified.

      Note: Do not modify any other part of the URL. The repository path and all other URL components must remain exactly as they are.

3. **Run the updated clone command**

a. In the dev container terminal, run the updated git clone command. For example:

   git clone ssh://git@devops-loop-control-ssh.devops-loop.svc.cluster.local:22/test_ts/.devopsconfig.git

b. The Control repository clones successfully because the dev container resolves the internal cluster-local address instead of attempting to reach the inaccessible external `service-` URL.

Code Genie

You can use **Code Genie** to automate tasks on the repository contents of DevOps Control.

Code Genie automates a wide range of tasks on the contents of a Control repository. You can trigger Code Genie either automatically through webhooks on page 212, or manually through the **Code Genie** button in the Plan user interface on page 158.

When a you create or update a PR in a repository, **Code Genie** launches a development container, clones the repository from the PR source branch, runs one or many commands, collects JavaScript Object Notation (JSON) results according to a standardized schema, maps them to PR comments, and can approve or reject the PR.

When you act on a work item in Plan, Code Genie launches a development container in the background, generates code based on the work item description, and creates a branch and pull request with the implementation.

Code Genie is a general-purpose feature for any PR automation task. Trivy Security Scanning on page 228, AI Code Review on page 224 and Automated JUnit Testing are some examples that uses Code Genie.

Default Capabilities

The following capabilities are available by default through the configuration file `.devops-loop/code-config.jsonc`, which is discussed in the **Configuration File** section.

- **Plan work item implementation**: When you act on a work item in Plan, Code Genie reads the work item coding instructions, generates code and creates a pull request implementing the requested changes. To use this feature, you must configure a coding CLI token for either GitHub Copilot or IBM Bob.
- **Trivy scanning and AI code review**: When you create or update a pull request, Code Genie automatically runs a Trivy security scan followed by an AI-powered code review using GitHub Copilot. The security scan runs independently of the coding CLI token. However, to run the code review, you must have a GitHub Copilot or IBM Bob token configured(based on what is configured is for code review)
- **Interact with Code Genie through PR comments**: When you add a comment to a pull request, you can ask Code Genie to explain specific code sections, suggest fixes, or answer questions about the implementation. Code Genie responds directly within the pull request comment thread.

You can use Code Genie for other use cases that include, but are not limited to the following tasks:

- Run unit tests on the PR source branch and reject the PR if any test fails.
- Let a large language model perform a code review for the PR and reject the PR if severe problems are found.

- Perform language-specific static code analysis, such as clang-tidy for C++ or ESLint for TypeScript, and reject the PR if critical problems are found. For more information, refer to Clang Tools 23.0.0 git documentation and ESLint.
- Automatically approve small PRs that only involve certain low-risk files, such as documentation files, possibly after performing sanity checks such as spell checking.

Code Genie uses JSON

JSON is the core format used by Code Genie to connect automated tools with pull request (PR) outcomes.

It defines a structured contract between:

- what runs inside the development container (tools, scripts, analyzers).
- what Code Genie displays or enforces in the PR (comments and approval decisions)

Code Genie uses JSON in two key areas: configuration and execution results, which is elaborated in the following sections:

Configuration File

When creating a Loop, a Control repo autoconfigures Code Genie via `.devops-loop/code-config.jsonc`.

This default configuration defines how Code Genie implements Plan work items, performs pull request security scanning and AI-powered code reviews, and responds to pull request comments.

```
{
  "code-genie": {
    "workitem": [
      {
        "dev-container": "Copilot",
        "coding-instructions": [/*URL to markdown file, for example
https://github.com/WDI-SEA/code-review/blob/master/code-guide.md*/],
        "coding-command": "export GH_TOKEN=${loop_property:copilotclitoken} &&
cp /opt/run-copilot.ts /usr/code/ && tsx /usr/code/run-copilot.ts --prompt-file=${promptFile}
--llm_args=\"--allow-all-tools --model claude-sonnet-4.5\""
      },
      {
        "dev-container": "Bob",
        "coding-instructions": [/*URL to markdown file, for example
https://github.com/WDI-SEA/code-review/blob/master/code-guide.md*/],
        "coding-command": "export BOBSHELL_API_KEY=${loop_property:bobtoken} &&
cp /opt/run-bob.ts /usr/code/ && tsx /usr/code/run-bob.ts --prompt-file=${promptFile}
--bob_args=\"--yolo --accept-license --auth-method api-key\""
      }
    ],
    "pull-request": [
      {
        "dev-container": "TrivyScan",
        "interactive": false,
        "custom-commands": [
          {
            "run-trivy": "cp /opt/run-trivy.sh /usr/code/ && chmod +x /usr/code/run-trivy.sh &&
REPO_URL=https://${platform-fqdm}/control/${repo-owner}/${repo}/src/branch/${pr-branch}/ /
```

```
usr/code/run-trivy.sh ${code-folder} ${result-json} --reject=high --trivyArgs=\"--scanners
 vuln,misconfig,secret,license --license-full --severity UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL\""
          }
        ]
      },
      {
        "dev-container": "Copilot",
        "interactive": true,
        "custom-commands": [
          {
            "code-review-command": "export GH_TOKEN=${loop_property:copilotclitoken} &&
 cp /opt/run-copilot.ts /usr/code/ && tsx /usr/code/run-copilot.ts --prompt-file=${promptFile}
 --llm_args=\"--allow-all-tools --model claude-sonnet-4.5\""
          }
        ]
      }/* Uncomment this block to use IBM Bob Shell for code reviews
      ,{
        "dev-container": "Bob",
        "interactive": true,
        "custom-commands": [
          {
            "code-review-command": "export BOBSHELL_API_KEY=${loop_property:bobtoken} &&
 cp /opt/run-bob.ts /usr/code/ && tsx /usr/code/run-bob.ts --prompt-file=${promptFile}
 --bob_args=\"--yolo --accept-license --auth-method api-key\""
          }
        ]
      }*/
    ]
  }
}
```

In the above example, the configuration splits into two sections: `"workitem"` , which defines how Code Genie implements Plan work items by selecting either the GitHub Copilot or IBM Bob entry based on the configured token, and `"pull-request"`, which defines what runs automatically on PR creation or update. Three entries run sequentially, TrivyScan (always active), GitHub Copilot code review (active by default), and IBM Bob code review (commented out). Setting `"interactive"` to false on TrivyScan runs it entirely in the background, while setting it to true on the AI code review containers allows developers to interact with Code Genie directly through PR comments.

The JSON file provides the following key properties:

`@workitem`

: This property specifies how Code Genie implements a work item from plan UI by defining the dev containers and the coding commands/instructions.

`@dev-container`

: This property specifies the name of the dev container to open. The dev container image should be accessible from Code Genie so that it can open the devcontainer. You must use a non-IDE dev container since such a dev container opens faster and occupies fewer resources than a dev container that contains an IDE. Since a Code

Genie dev container runs in the background, without user interaction, including an IDE would be unnecessary overhead. For more information, refer to the non-IDE dev container on page 209.

`@coding-instructions`

This property specifies an array of URLs pointing to markdown files that contain the coding guidelines Code Genie passes to the AI as context. Replace the placeholder comment with one or more URLs to your own coding standards documents.

`@coding-command`

This property specifies how Code Genie invokes the AI coding assistant to implement a work item. It sets the required authentication token, copies the assistant's runner script into the working directory, and executes the script with a prompt file containing the work item details. Additional arguments can also be provided to configure the model and tool permissions used during code generation.

`@pull-request`

This property specifies how Code Genie is triggered. Code genie is triggered on PR creation and when more commits are added to the PR.

`@custom-commands`

This is an array of commands to run in the dev container. They run sequentially in the order specified in the JSON file. The current working directory for each command invocation is the folder in the dev container file system where the Control repository was cloned. Each command is represented by a JSON object in the array, with a single property describing what the command does (for example, "run-trivy"). The value of the property is the actual Unix command-line to invoke in the dev container. It can use any tool or script that is present in the dev container.

`@interactive`

This property if set to true allows users to interact with Code Genie through PR comments for AI dev containers.

By default, Trivy scanning and code review run on every pull request. To turn off these checks, see Disabling Trivy Scanning and Code Review on Pull Requests on page 206.

If you use Code Genie in an air-gapped environment, you must disable Copilot code review. See Disabling AI Code Review in an Air-Gapped Environment on page 207.

Variables

Before Code Genie runs a command, it replaces certain variables with actual values:

`${platform-fqdn}`

Fully qualified domain name of the Loop installation.

`${repo-owner}`

Owner of the cloned repository (typically a teamspace).

`${repo}`

Name of the cloned repository.

`${pr-branch}`

Name of the PR source branch (currently checked out).

`${code-folder}`

Folder in the dev container where the Control repo is cloned.

`${result-json}`

Path to JSON results file (unique name like cg-result-10-1.json). An invoked command might choose to generate this file, and its contents then must comply with the standardized schema for PR comment translation.

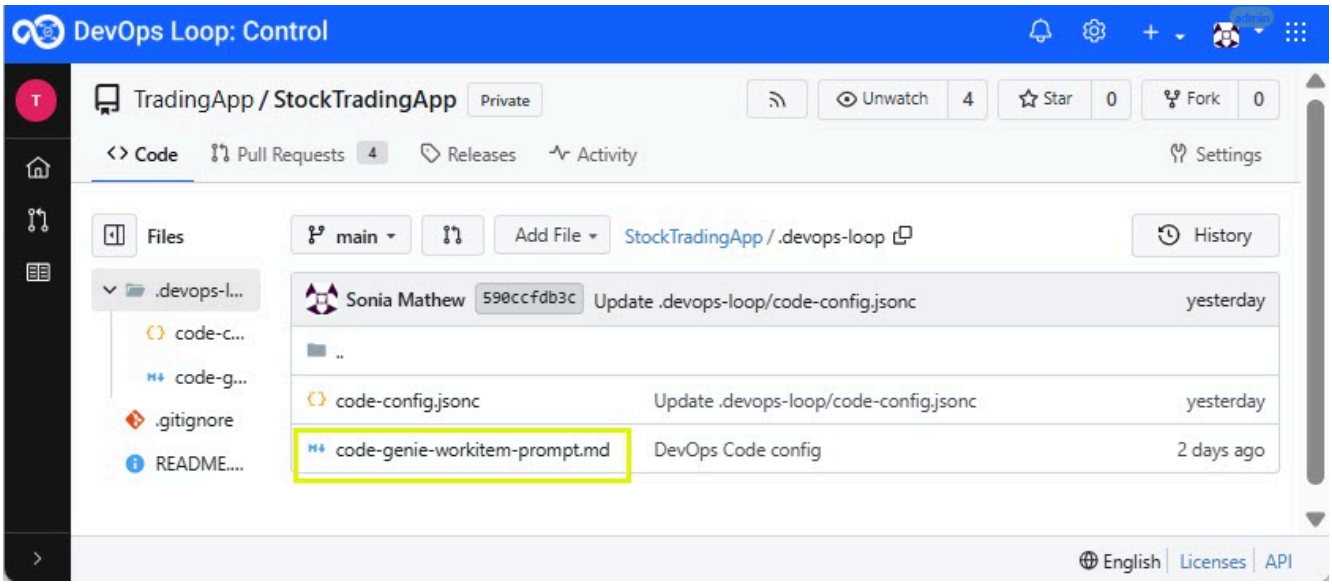
`${promptFile}`

Reserved runtime placeholder used by the Code Genie backend. It is not a real file path and must not be modified or interpreted in the configuration. During execution, the backend replaces it with the actual generated prompt file that contains the work item context. This ensures the coding command receives the correct input without requiring any manual handling.

Note: You can use comments (both line comments (//) and block comments (/* */)) in the configuration file.

Prompt file

When a loop is created, a new prompt file also gets automatically created in the Control repo: `.devops-loop/code-genie-workitem-prompt.md`, that is, at the same location where `code-config.jsonc`. is created.



This prompt is helpful when you use the code genie feature in a Plan work item to implement a work item. You can customize it, if needed.

Note: You must not modify placeholders such as ${targetDir} in the `code-genie-workitem-prompt.md`, as doing so may lead to unexpected behavior or results.

Using webhooks and runner scripts

Each command-related property like `"custom-commands"` or `"coding-command"` in the configuration file invokes runner scripts that are executed inside a dev container by Code Genie to automate tasks such as implementing Plan work items, handling pull request workflows and supporting other customized usecases. For more information, see Runner Scripts on page 210.

In addition to the configuration file, Code Genie requires a webhook to be present on the Control repository. For a repository that is created as part of loop creation, the webhook is automatically created. If you create a repository manually, you must create the webhook. See Creating webhooks on page 212.

Coding CLI token configuration

For Plan work item implementation and AI code review to function, a coding CLI token must be configured. For more information, see Configuring Coding CLI token on page 215.

Execution Result File

The result of running a command can be stored in a JSON result file. The variable `${result-json}` expands to the full path of that JSON file. The file name is unique for each command invocation and includes the Control PR number and index within the custom-commands array. For example, the file name can be **cg-result-10-1.json** if the Control PR number is 10 and the command is the first command in the custom-commands array.

After running the command, Code Genie checks whether a JSON result file is available. If present, it must comply with a standardized JSON schema. The JSON properties are as follows:

`@results`

This is an array of objects that specify code locations, typically locations within files of the Control repository. Each such object is mapped to a PR comment that is associated with a specific file ("path" property) and a line within that file ("new_position" property). The PR comment text is specified by the "body" property and might use markdown for formatting.

`@pr-comments`

This is an array of strings, each of which is mapped to a PR comment that is not associated with any file. The strings might use markdown for formatting.

`@review`

This object can be present to specify whether Code Genie should make a review of the PR as either approved (property "approved" set to true) or rejected (property "approved" set to false). In either case, you can set the "comment" property to a comment explaining the PR approval or rejection.

The following example shows the results JSON file generated by Code Genie when it specifically runs a Trivy scan.

Note: The format is the same when Code Genie runs a Copilot AI code review, but the contents can be different and the "review" property is not used.


```
{
  "results": [
    {
      "path": "package-lock.json",
      "new_position": 1,
      "body": "[VULNERABILITY] CVE-2024-29041 | MEDIUM | express (v4.16.0) | Fix: 4.19.2,
 5.0.0-beta.3"
    },
    {
      "path": "package-lock.json",
      "new_position": 1,
      "body": "[VULNERABILITY] CVE-2024-43796 | LOW | express (v4.16.0) | Fix: 4.20.0, 5.0.0"
    },
    {
      "path": "package-lock.json",
      "new_position": 1,
      "body": "[LICENSE] MIT | LOW | Pkg: express"
    }
  ],
  "pr-comments": [
    "Started: 2026-05-13T15:28:24Z, Ended: 2026-05-13T15:28:33Z",
    "Findings summary => vulnerabilities: 2, misconfigurations: 0, secrets: 0, licenses: 1, total:
 3",
    "Full repository analysis performed. Total number of files scanned: 4, Total files with
 findings: 1. Local changes detected: 1",
    "",
    "✅ Security scan passed (No issues at HIGH or CRITICAL).",
    "🛡️# Detailed Security Analysis",
    "| File | Line | Type | ID | Severity | Package | Details |",
    "| :--- | :--- | :--- | :--- | :--- | :--- | :--- |",
    "|
 [package-lock.json](<https://10.83.92.71.nip.io/control/test_ts/testloop/src/branch/testb5/packa
ge-lock.json>) | 1 | Vulnerability | CVE-2024-29041 | MEDIUM | express | cause malformed URLs to be
 evaluated |",
    "|
 [package-lock.json](<https://10.83.92.71.nip.io/control/test_ts/testloop/src/branch/testb5/packa
ge-lock.json>) | 1 | Vulnerability | CVE-2024-43796 | LOW | express | Improper Input Handling in
 Express Redirects |",
    "|
 [package-lock.json](<https://10.83.92.71.nip.io/control/test_ts/testloop/src/branch/testb5/packa
ge-lock.json>) | 1 | License | MIT | LOW | express | Review Compliance |"
  ],
  "review": {
    "approved": true,
    "comment": "All security checks passed threshold."
```

```
  }
}
```

Related information

Create a Loop on page 140

Disabling Trivy Scanning and Code Review on Pull Requests on page 206

Disabling Code Review in an Air-Gapped Environment on page 207

Disabling both Trivy Scanning and AI Code review

You can disable the Trivy security scan and AI-powered code review that Code Genie runs automatically on pull request creation or update.

About this task

By default, Code Genie runs a Trivy security scan and an AI-powered code review every time a pull request is created or updated. To turn off these checks, comment out the corresponding entries in the `"pull-request"` array of `.devops-loop/code-config.jsonc`.

1. Open `.devops-loop/code-config.jsonc` in the Control repository.
2. Locate the `"pull-request"` array.
3. Enclose the `TrivyScan` entry and the `Copilot` entry in block comments (`/* */`).
4. Save the file.

Results

With no active entries in the `"pull-request"` array, Code Genie does not run a security scan or a code review when a pull request is created or updated.

Note: Disabling Trivy scanning and code review affects only the `"pull-request"` array. It does not affect the Plan work item implementation (`"workitem"` array).

Example

The following example shows the `"pull-request"` array with Trivy scanning and code review disabled:

```
"pull-request": [
    /*{
      "dev-container": "TrivyScan",
      "interactive": false,
      "custom-commands": [
        {
          "run-trivy": "cp /opt/run-trivy.sh /usr/code/ && chmod +x /usr/code/run-trivy.sh &&
 REPO_URL=https://${platform-fqdm}/control/${repo-owner}/${repo}/src/branch/${pr-branch}/ /
usr/code/run-trivy.sh ${code-folder} ${result-json} --reject=high --trivyArgs=\"--scanners
 vuln,misconfig,secret,license --license-full --severity UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL\""
```

```
        }
      ]
    },
    {
      "dev-container": "Copilot",
      "interactive": true,
      "custom-commands": [
        {
          "code-review-command": "export GH_TOKEN=${loop_property:copilotclitoken} &&
cp /opt/run-copilot.ts /usr/code/ && tsx /usr/code/run-copilot.ts --prompt-file=${promptFile}
--llm_args=\"--allow-all-tools --model claude-sonnet-4.5\""
        }
      ]
    }*/
    /* Uncomment this block to use IBM Bob Shell for code reviews
    ,{
      "dev-container": "Bob",
      "interactive": true,
      "custom-commands": [
        {
          "code-review-command": "export BOBSHELL_API_KEY=${loop_property:bobtoken} &&
cp /opt/run-bob.ts /usr/code/ && tsx /usr/code/run-bob.ts --prompt-file=${promptFile}
--bob_args=\"--yolo --accept-license --auth-method api-key\""
        }
      ]
    }*/
  ]
```

Related information

Code Genie on page 199

Disabling AI Code review in Air-gapped environment

In an air-gapped environment, you must disable the AI-powered code review that Code Genie runs on pull request creation or update.

About this task

An air-gapped environment is a deployment that is physically or logically isolated from the public internet, with no outbound network access to external services.

The `Copilot` code review script requires internet access to communicate with the GitHub Copilot service. In an air-gapped environment, this communication is not possible. You must comment out the `Copilot` entry in the `"pull-request"` array of `.devops-loop/code-config.jsonc` before you use Code Genie in an air-gapped environment.

1. Open `.devops-loop/code-config.jsonc` in the Control repository.
2. Locate the `"pull-request"` array.
3. Enclose the `Copilot` entry in block comments (`/* */`).

Leave the `TrivyScan` entry active as it does not require internet access and can continue to run on pull request creation and update.

4. Save the file.

Results

With the `Copilot` entry disabled, Code Genie continues to run the Trivy security scan on pull request creation and update, but no longer attempts to run AI-powered code review.

Attention: Disabling `Copilot` code review is required in air-gapped environments, not optional. If the `Copilot` entry is left active, the code review dev container fails because it cannot reach the GitHub Copilot service.

Example

The following example shows the `"pull-request"` array configured for an air-gapped environment, with `TrivyScan` active and `Copilot` code review disabled:

```
"pull-request": [
    {
      "dev-container": "TrivyScan",
      "interactive": false,
      "custom-commands": [
        {
          "run-trivy": "cp /opt/run-trivy.sh /usr/code/ && chmod +x /usr/code/run-trivy.sh &&
 REPO_URL=https://${platform-fqdm}/control/${repo-owner}/${repo}/src/branch/${pr-branch}/ /
usr/code/run-trivy.sh ${code-folder} ${result-json} --reject=high --trivyArgs=\"--scanners
 vuln,misconfig,secret,license --license-full --severity UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL\""
        }
      ]
    }
    /*,{
      "dev-container": "Copilot",
      "interactive": true,
      "custom-commands": [
        {
          "code-review-command": "export GH_TOKEN=${loop_property:copilotclitoken} &&
 cp /opt/run-copilot.ts /usr/code/ && tsx /usr/code/run-copilot.ts --prompt-file=${promptFile}
 --llm_args=\"--allow-all-tools --model claude-sonnet-4.5\""
        }
      ]
    }*/
  ]
```

What to do next

Installing extensions in dev containers in an air-gapped environment

When a custom dev container, such as the Java devcontainer, is launched in an air-gapped environment, extensions do not get installed by default, with the exception of the DevOps Code extension itself. This occurs because extension installation

normally requires outbound access to a public extension marketplace, which is not available in an air-gapped environment. In order to make the required extensions available, you can follow the steps below:

1. Identify the extensions required by the custom devcontainer. For example, the Java dev container requires the extensions used for Java development and debugging.
2. Add the required extensions to the extension registry. Refer to extension registry on page 187 documentation.
3. Launch the custom devcontainer, for example the Java dev container.
4. Install the required extensions manually inside the launched dev container, following the same extension registry documentation.

The required extensions are then available inside the devcontainer, in addition to the default DevOps Code extension.

Related information

Code Genie on page 199

Disabling Trivy Scanning and Code Review on Pull Requests on page 206

Non-IDE dev container

You can configure a non-IDE dev container to run background tasks, such as CI/CD pipelines or Code Genie automation, without the overhead of an IDE.

A **non-IDE dev container** runs as a plain container without launching an IDE. It launches faster and occupies less resources than a standard dev container, making it suitable for background tasks that run without user interaction - such as PR automation, CI/CD pipelines, or CLI-only workloads, where including an IDE would be unnecessary overhead.

You can configure one by setting the `"ide": "false"` property in your devcontainer.json file. The following devcontainer.json file on page 172 shows a sample configuration for a non-IDE dev container:

```
{
  "name": "My Dev Container",
  "description": "A plain development container without IDE support.",
  "build": {
    "dockerfile": "Dockerfile"
  },
  "ide": "false",
  "image": "<username>/<container-name>:<tag>"
}
```

You can use a non-IDE dev container with Code Genie on page 199 to improve performance. When a PR is created, Code Genie launches a dev container in the background. Excluding the IDE minimizes startup time and resource consumption, resulting in faster, more lightweight automation.

The following Dockerfile on page 172 shows a sample configuration for a non-IDE dev container with the IDE installation removed:

```
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && \
    apt-get install -y \
        sudo \
        curl \
        git \
        ca-certificates \
        build-essential && \
    apt-get clean

RUN chmod -R ug+w /etc/pki/ca-trust/source /etc/pki/ca-trust/extracted || true

RUN useradd -m -u 1001 devuser
USER 1001
WORKDIR /home/1001
CMD ["tail", "-f", "/dev/null"]
```

Configuring the timeout for non-IDE dev containers

DevOps Loop automatically terminates idle non-IDE dev containers if not in use after a configurable timeout period.

The timeout value is specified in seconds and can be configured during installation.

For example, to set the timeout to 86,400 seconds (24 hours), add the following to the Loop installation script on page 69:

```
NON_IDE_CONTAINER_TIMEOUT=86400

if [ -n "${NON_IDE_CONTAINER_TIMEOUT}" ]; then
 HELM_OPTIONS="${HELM_OPTIONS} --set global.nonIdeContainerTimeout=${NON_IDE_CONTAINER_TIMEOUT}"
fi
```

Runner Scripts

You can use runner scripts to define the core execution logic that Code Genie runs inside a dev container when a pull request is created or updated.

When **Code Genie on page 199** triggers a container, it runs a command string defined in `code-config.jsonc` via the `custom-commands` or `coding-command` properties. The command string invokes the runner script, which executes the task, processes the results, and generates the structured JSON output that Code Genie reads to post PR comments and decide on approval or rejection.

The default Code Genie configuration includes built-in runner scripts that you cannot access or modify. You can use the information in this topic to create your own runner script for custom use cases, such as running unit tests or performing static code analysis.

The following table lists the built-in runner scripts used in the default Code Genie configuration:

Script	Type	Purpose
`run-trivy.sh`	Shell script	Runs Trivy security scanning on the PR source branch and outputs findings to the result JSON file.
`run-copilot.ts`	TypeScript script	Invokes GitHub Copilot to implement work items or perform AI code reviews on PRs.
`run-bob.ts`	TypeScript script	Invokes IBM Bob to implement work items or perform AI code reviews on PRs.

You can write your own runner script in any language that the dev container supports. For a step-by-step walkthrough of creating a custom runner script, see DevOps Loop: Automated JUnit Testing with Code Genie.

What a Runner Script Does

Regardless of the use case, a runner script performs the following functions:

- Executes the core task, such as performing a security scan, invoking an AI code review, or running tests.
- Generates a structured JSON result file that Code Genie reads to post PR comments and determine approval or rejection.

Creating a Custom Runner Script

To create a custom runner script, you must embed the script into the dev container image and ensure that the command in `code-config.jsonc` passes the inputs that the script expects.

Embedding the script in the dev container: You can embed a runner script into the dev container image at build time by using the `COPY` instruction in the Dockerfile, followed by `chmod +x` to make it executable. The following Dockerfile shows a minimal example that embeds a JUnit runner script into the dev container image:

```
FROM registry.access.redhat.com/ubi9
# Install Java, Maven, Git, and JQ in a single layer
RUN dnf install -y --allowerasing --setopt=install_weak_deps=False \
    git java-17-openjdk java-17-openjdk-devel maven jq && dnf clean all
ENV JAVA_HOME=/usr/lib/jvm/java-17-openjdk
ENV PATH="${JAVA_HOME}/bin:${PATH}"
# Configure non-root user and directories
RUN useradd -u 1001 -m codeuser && \
    mkdir -p /opt /tmp/junit-results && \
    chown -R 1001:1001 /opt /tmp/junit-results /home/codeuser
COPY run-junit.sh /opt/run-junit.sh
RUN chmod +x /opt/run-junit.sh
USER 1001
WORKDIR /home/codeuser
CMD ["/bin/bash"]
```

The script is then available inside the container at a known path. The command in `code-config.jsonc` copies it to the working directory before executing it. Because the Dockerfile bakes the script into the image, the container image and the

command string are tightly coupled, the image determines what scripts are available, and the configuration determines how it invokes them. Any mismatch between the two causes the automation to fail. For more information, see Creating Dev Containers using a Dockerfile on page 172.

Mapping commands to script expectations: The commands in `code-config.jsonc` must map directly to what the runner script expects. Common inputs include:

- `${code-folder}`: The working directory inside the container where Code Genie clones the repository and executes commands.
- `${result-json}`: The target path where the script writes its execution summary in JSON format for Code Genie to parse.
- Use-case specific flags: Such as rejection thresholds or tool-specific arguments that control how the script evaluates results and determines PR approval or rejection.

The following code-config.jsonc shows how the Trivy runner script is invoked, with the required inputs passed as part of the command string:

```
{
  "code-genie": {
    "pull-request": {
      "dev-container": "TrivyScan",
      "custom-commands": [
        {
          "run-trivy": "cp /opt/run-trivy.sh /usr/code/ && \
                        chmod +x /usr/code/run-trivy.sh && \

 REPO_URL=https://${platform-fqdm}/control/${repo-owner}/${repo}/src/branch/${pr-branch}/ \
                        /usr/code/run-trivy.sh ${code-folder} ${result-json} \
                        --reject=high \
                        --trivyArgs=\"--scanners vuln,misconfig,secret,license --license-full
 --severity UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL\""
        }
      ]
    }
  }
}
```

Creating webhooks

For manually-created Control repositories, you must create a webhook so that Code Genie can detect pull request activity and trigger automated scans and reviews

Before you begin

Before you begin, ensure that you have:

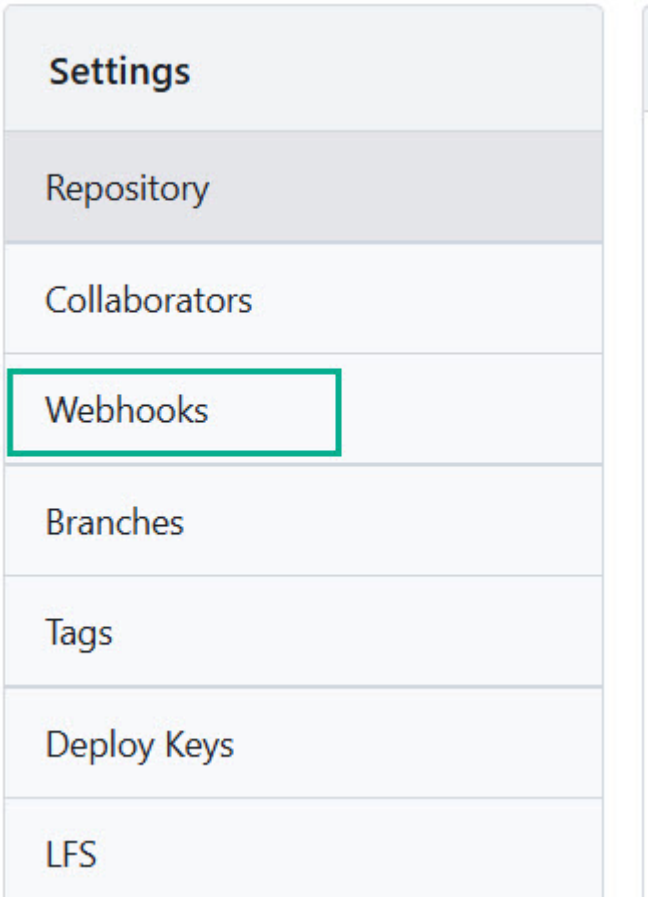
- Administrator access to the **Control** repository
- Created a Control repository manually.
- Read Code Genie on page 199 documentation.

About this task

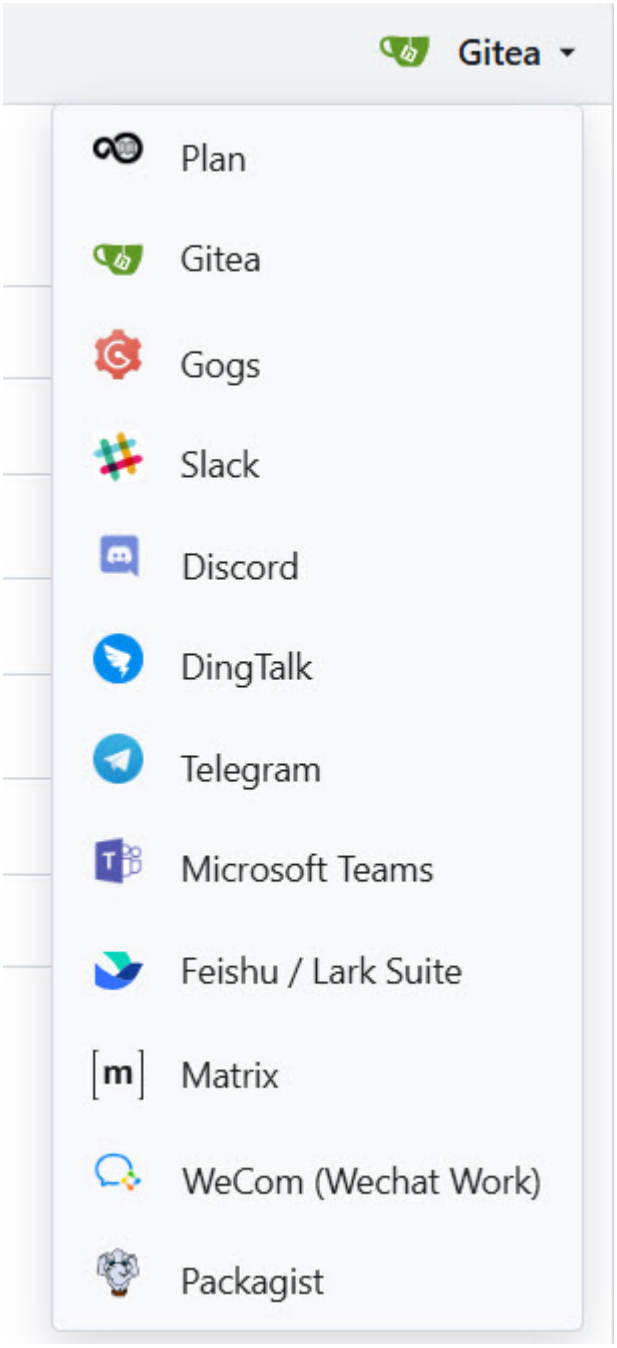
A webhook is an HTTP callback that Gitea triggers automatically when repository events occur. When an event matches your configured trigger, Gitea sends a POST request with a JSON payload to the **Code Genie** target URL. You can filter triggers by event type and branch pattern, giving you precise control over which pull request activities notify Code Genie.

Important: Automatic configuration of the webhook only applies to Control repositories created through the Loop creation process. If you create a Control repository manually, you must create the webhook yourself, as described in this procedure. You must also configure branch protection rules and create the `.devops-loop/code-config.jsonc` file for Code Genie to work correctly.

1. Navigate to your Control repository in Gitea and click **Settings**.



2. In the left navigation panel, click **Webhooks** under **Settings**.
3. Click **Add Webhook** and select **Gitea** from the drop-down list.



4. Configure the webhook with the following values.
    - Target URL: `http://devops-code-container-mgr:3005/api/pull-request`
    - HTTP Method: **POST**
    - POST Content Type: `application/json`
5. Under **Trigger On**, select **Custom Events**.
6. In the **Pull Request Events** section, select the following check boxes.
    - **Pull Request** — fires when a pull request is opened, closed, reopened, or edited.
    - **Pull Request Synchronized** — fires when new commits are pushed to an open pull request.

7. In the **Branch Filter** field, enter `*` to apply the webhook to all branches.
   To restrict to specific branches, use a glob pattern such as `master` or `{master,release*}`.
8. Click **Add Webhook** to save the configuration.
9. Confirm that the webhook appears in the **Webhooks** list with a green status indicator.

Results

You have created a webhook on the Control repository. Gitea will now send a POST request to Code Genie each time a pull request is opened or synchronized on the repository.

What to do next

After creating the webhook, you must complete the remaining manual configuration steps before Code Genie can scan pull requests:

- Configure branch protection rules for the main branch.
- Create the `.devops-loop/code-config.jsonc` file in the Control repository to enable Trivy scanning and AI code review with CoPilot.

Configuring Coding CLI token

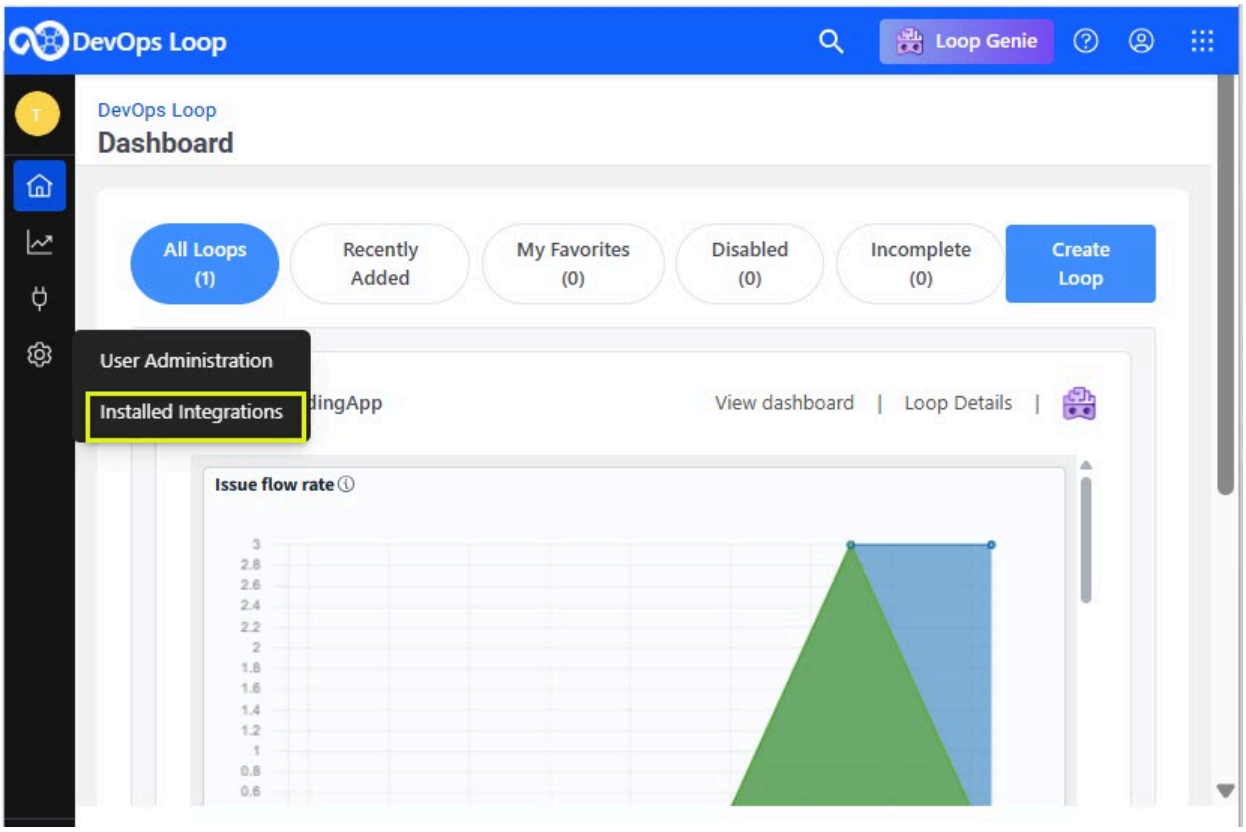
Configure a coding CLI token for IBM Bob or GitHub Copilot to enable Code Genie automated work item implementation and AI-powered code reviews.

About this task

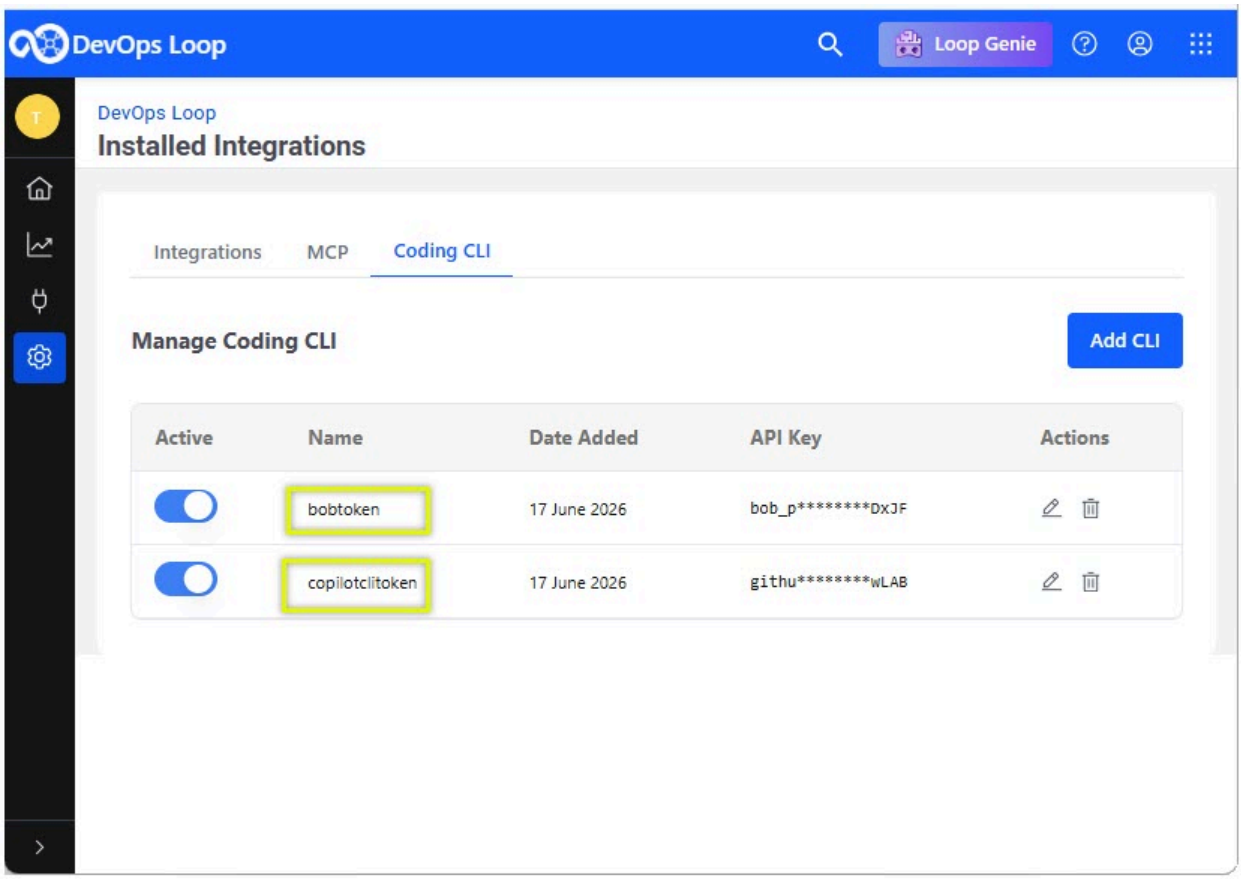
For work item implementation and AI code review to function, **Code Genie** requires a Coding CLI token. By default, Code Genie supports two AI providers: GitHub Copilot (using built-in dev container) and IBM Bob (dev container using IBM Bob Shell is available on DockerHub).

The following steps guide you through creating a Coding CLI token:

1. Generate a token or API key from your preferred AI provider:
   Choose from:
   - **GitHub Copilot:** You can obtain a GitHub personal access token with the appropriate Copilot permissions. To create your token, navigate directly to your GitHub Personal Access Tokens Settings. For detailed documentation on managing these personal access tokens to configure Code Genie capabilities, refer to Managing your personal access tokens. If access is denied, contact your IT team to request configuration and enablement of valid GitHub Copilot subscription and username.
   - **IBM Bob:** You can generate an API key from the IBM Bob admin page. Refer to the IBM Bob API Keys Documentation.
2. Navigate to the **Manage Coding CLI** menu in the **Coding CLI** tab from **Installed Integrations** in the **DevOps Loop** settings.


DevOps Loop
Loop Genie
DevOps Loop
Dashboard
All Loops (1)
Recently Added
My Favorites (0)
Disabled (0)
Incomplete (0)
Create Loop
User Administration
Installed Integrations
View dashboard | Loop Details |
Issue flow rate

3. Add the generated key or token to the coding CLI configuration section and save the entry.



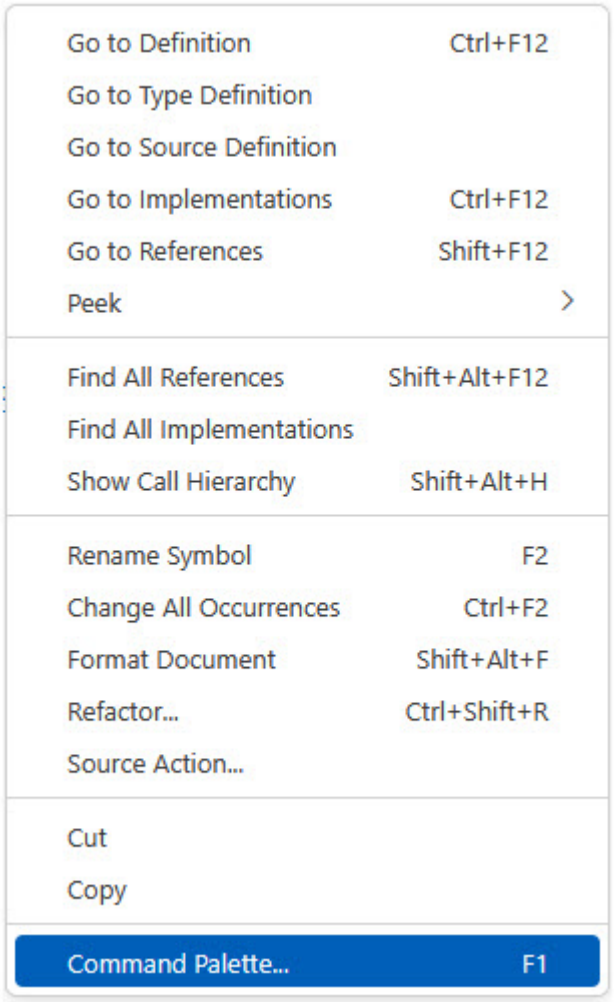
Creating Plan work items from Code

The DevOps Code extension allows integration with HCL DevOps Plan (Plan), enabling you to create and manage work items directly from your development environment in the browser IDE.

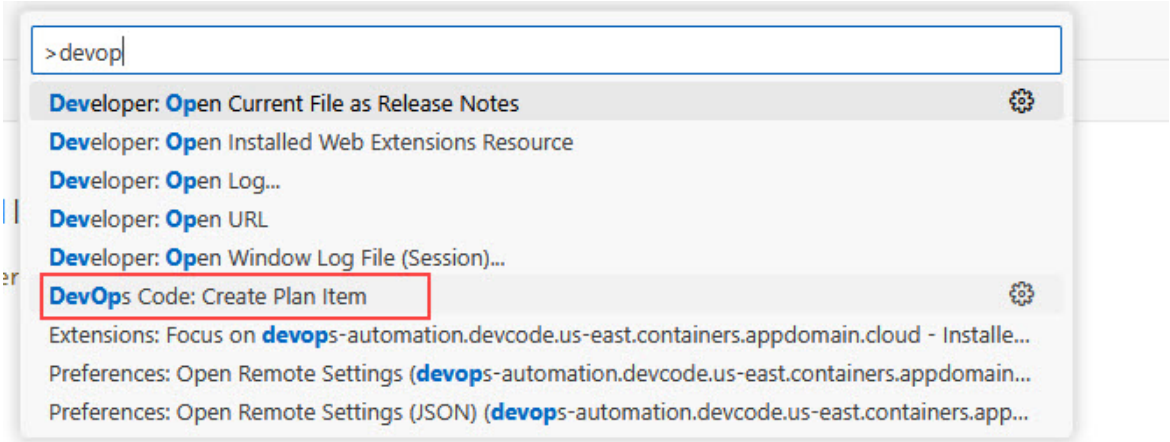
About this task

You can follow these steps to configure DevOps Code extension in the Code browser IDE, and create work items in HCL DevOps Plan (Plan).

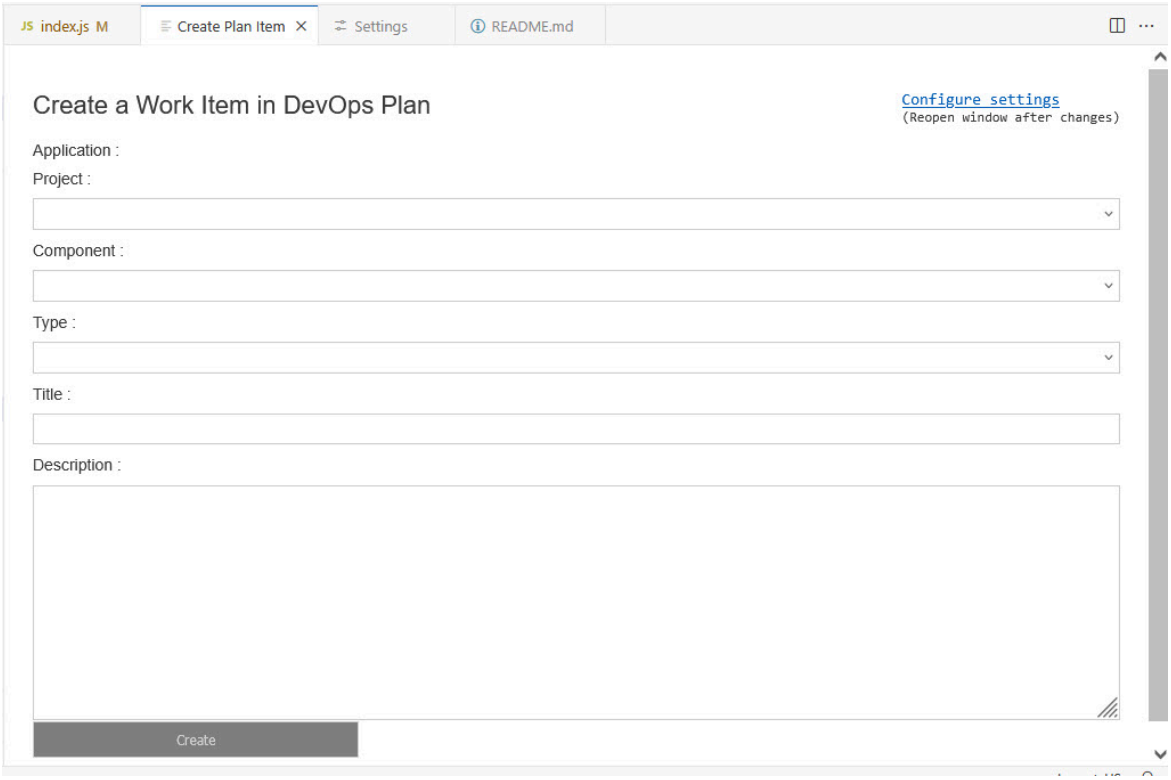
1. Open the Command Palette.
   Right-click to open the context menu and select **Command Palette** in the editor.



2. Access the DevOps Code Command.
    a. Choose the Code command from the list of available commands.



The **Create Plan Item** window is displayed.



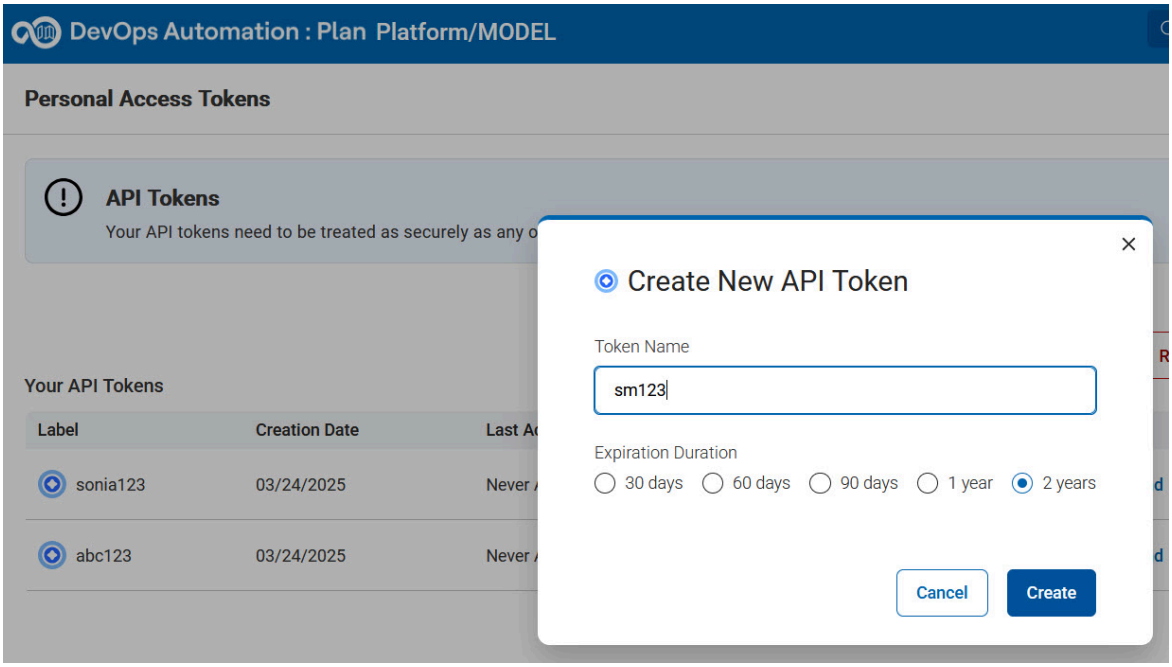
b. Click the **Configure settings** button to integrate DevOps Code with Plan for the first time.

   The **Settings** window is displayed to add the Personal Access Token, Plan Server URL, and Team Space ID.

   You can add these fields by navigating to the Plan platform, which is detailed in the next step.

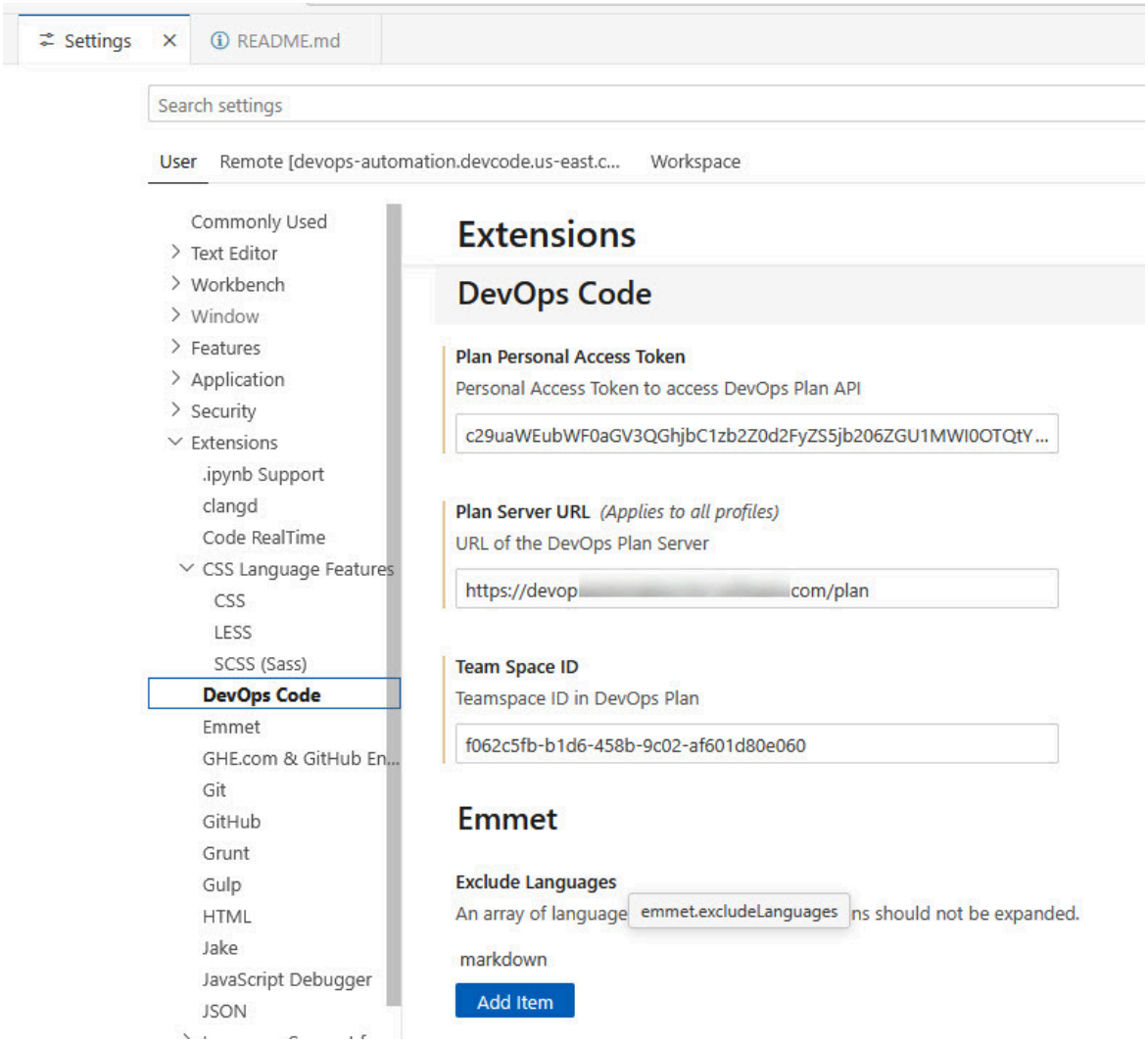
3. Perform the following steps to get configuration details from Plan:

a. Click the **Plan** button in the HCL DevOps Loop switcher to navigate to the Plan page.
b. Go to the **Settings** section of Plan, and enter the **Personal Access Token** (or API tokens) in the **Create New API Token** dialog.



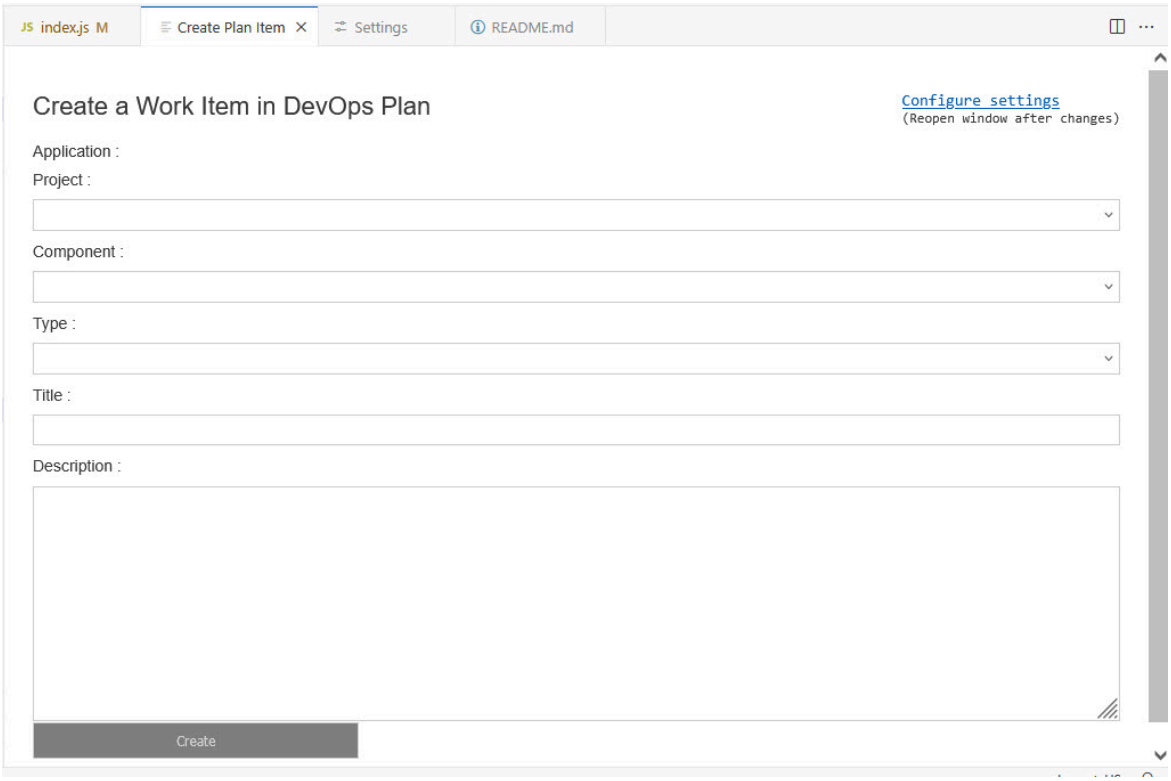
4. Perform the following steps to enter the configuration details:
   a. Navigate back to the Code application window from the switcher.
   b. Update the configuration details in the **Settings** window of Code.

Note: To find the team space ID to use, open a work item in Plan and look in the browser address bar for the text between the words "repos" and "databases". For example, 'f062c5fb-b1d6-458b-9c02-af601d80e060' in https://<DevOpsLoopURL>/plan/#/ccmweb/view/repos/f062c5fb-b1d6-458b-9c02-af601d80e060/databases/MODEL/records/WorkItem/MODEL00000179

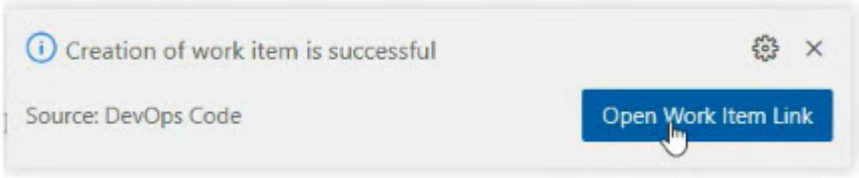


5. Create a work item:
    a. Select the **DevOps Code** command from **Command Palette**.

    The **Create Plan Item** window is displayed again.

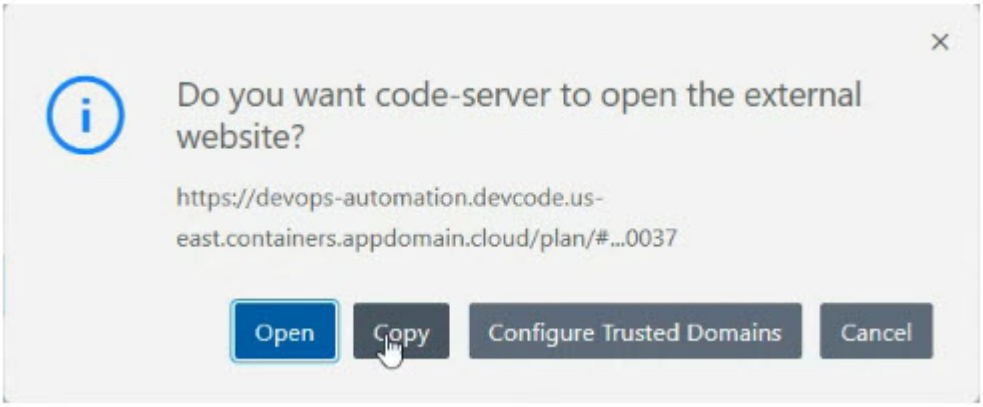


b. Fill in the required fields such as Application Project, Component, Type, Title, and Description.
c. Click the **Create** button to create your work item in Plan.



A prompt is displayed at the end of the IDE to indicate that the work item has been created successfully.

6. Click **Open Work Item Link** to view the work item on the Plan page.
   A dialog is displayed to open the Plan web page.



7. Click **Open** to navigate to the work item page on the Plan web page.

Results

By following these steps, you can easily configure the Code extension and begin creating and managing work items directly within your development environment. This integration eliminates the need to switch between your Code IDE and Plan.

Note: Currently, the DevOps Code extension has the following limitations:

- At most one Plan application can be used. If you have multiple applications, the first one will be used.
- The Plan application must use the Agile business flow. Other flows are currently not supported.
- Your Plan project(s) must have at least one component.

Installing GitHub Copilot in dev containers

As an admin, you can install the GitHub Copilot extension to enable AI-powered code suggestions, and chat assistance within your dev container.

Before you begin

You must have performed the following tasks:

- Gained access to edit the .devopsconfig control repository.
- Have a valid GitHub account with an active GitHub Copilot subscription.

About this task

This task describes how you can configure a dev container to support GitHub Copilot, and install and activate the extension within the running environment.

1. Configure the dev container image:
    a. Navigate to the .devopsconfig Control repository.
    b. Edit the existing devcontainer.json file (for example, within the Java dev container configuration), and update the image property to define the specific container version:

```
"image": "baravich/devcontainer-java-copilot:2.0.000"
```

c. Commit and push the changes to the repository.

2. Launch the dev container:
   a. In Code, locate the appropriate tile for the dev container configured in Step 1.
   b. Click the tile to launch the dev container.
3. Install the Copilot Extensions:
   a. You must run a script to install the necessary binaries once the environment is running. Open a new Terminal in the dev container environment.
   b. Copy and paste the following command to download and run the installation script:

```
curl -fsSL
 https://raw.githubusercontent.com/sunpix/howto-install-copilot-in-
code-server/refs/heads/main/install-copilot.sh
```

   This script automatically installs both the GitHub Copilot and Copilot Chat extensions.
4. Authenticate and access the Copilot:
   a. Locate the GitHub Copilot icon in the activity bar or status bar, once the installation is complete.
   b. Select the icon and follow the prompts to sign in to your GitHub account.
   c. Authorize the extension to access your subscription.

Results

The GitHub Copilot and Chat extensions are now installed and active. You can now use AI assistance for code generation and technical questions directly within your development environment.

Control

Control manages source code within the DevOps Loop using GitHub compatible APIs that integrate with existing developer tools, workflows, and automation. It supports version management, collaboration, and code governance as development progresses.

For more details refer to the Control Documentation.

AI code reviewing with Code Genie

A Control repository that is created as part of creating a Loop will be automatically configured to enable AI-assisted code review using **Code Genie** with the Copilot dev container. Use this procedure when you want to enable interactive, AI-powered code suggestions on pull requests across your team.

Before you begin

Before you begin, ensure that you have:

- Administrator access to your team space.
- Created a teamspace. See Creating a teamspace on page 132.

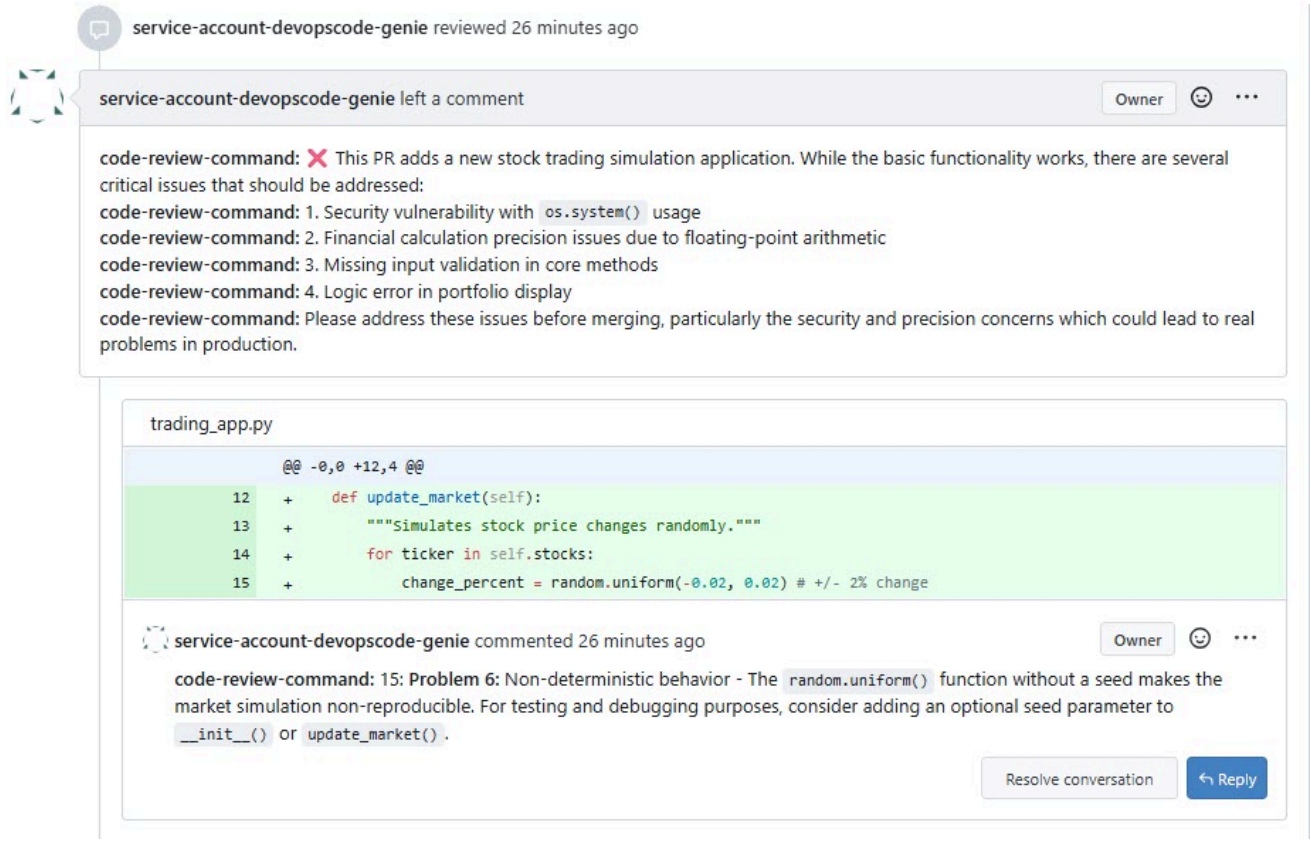
- Created a **DevOps Loop** so that the Control repository, webhook, branch protection, and default `.devops-loop/code-config.jsonc` are provisioned automatically. See Creating a loop on page 140.
- Read the Code Genie on page 199 topic to understand how Code Genie works with AI-integrated features.
- Obtained GitHub Copilot or an IBM Bob AI agent token and save it under Manage Coding CLI on page 215 section. This token is required for the Copilot or Bob dev container to authenticate with the AI service.
- Ensured your team follows a branching strategy on page 233 with a protected main branch.

Attention: If you are deploying in an air-gapped environment, disable the Copilot code review. The Copilot script requires internet access to communicate with the GitHub Copilot service, which is not available in an air-gapped environment. See Disabling Code Review in an Air-Gapped Environment on page 207.

About this task

Code Genie runs the `Copilot` dev container on each pull request to perform AI-assisted code reviews. When a pull request is opened, Code Genie (`service-account-devopscode-genie`) launches an AI agent using the configured model, such as `claude-sonnet-4.5`, and analyzes the pull request diff in the context of the repository.

During the review process, the AI agent reads the pull request diff, examines modified files, checks for repository-specific guidance in `.github` or `.agents` folders, and evaluates the changes for potential security vulnerabilities, logic defects, input validation issues, edge cases, maintainability concerns, and overall code quality. Review findings are posted directly to the pull request as both summary comments and line-level annotations.



Code Genie also provides a detailed CLI execution log that records the review workflow, including prompt generation, repository inspection, file reads, tool usage, code analysis steps, output generation, and validation. These logs provide transparency into how the AI agent arrived at its review findings and can be used for troubleshooting and audit purposes.

../videos/cli-output.mp4

Important: When a pull request is closed, the `Copilot` dev container associated with it is automatically terminated. Any other open pull requests waiting for Code Genie results at that point may not receive their comments because the container is no longer running. This is a temporary limitation in the current release.

1. Open `.devops-loop/code-config.jsonc` in the **Control** repository.
2. Replace the existing configuration with the following, which enables the Copilot AI code generation container:

```
{
  "code-genie": {
    "pull-request": [
      {
        "dev-container": "Copilot",
        "interactive": true,
        "custom-commands": [
          {
```

```
            "code-review-command": "export GH_TOKEN=${loop_property:copilotclitoken} &&
cp /opt/run-copilot.ts /usr/code/ && tsx /usr/code/run-copilot.ts --prompt-file=${promptFile}
--llm_args=\"--allow-all-tools --model claude-sonnet-4.5\""
          }
        ]
      }

      /* Uncomment this block to use IBM Bob Shell for code reviews
      ,
      {
        "dev-container": "Bob",
        "interactive": true,
        "custom-commands": [
          {
            "code-review-command": "export BOBSHELL_API_KEY=${loop_property:bobtoken} &&
cp /opt/run-bob.ts /usr/code/ && tsx /usr/code/run-bob.ts --prompt-file=${promptFile}
--bob_args=\"--yolo --accept-license --auth-method api-key\""
          }
        ]
      }
      */
    ]
  }
}
```

You can adjust the following parameters for the `Copilot` container:

`interactive`

Set to `true` to enable communication of code-genie with user using Pull Request comments.

`loop_property:copilotclitoken`

The Loop property key that holds your Copilot CLI authentication token. Ensure this property is set on the Loop before enabling this container.

`--prompt-file`

Path to the prompt file that instructs the AI model what to generate or review. The `${promptFile}` variable is resolved at runtime by Code Genie.

`--model`

The AI model to use for code generation. Defaults to `claude-sonnet-4.5`. You can substitute another supported model as required by your organization's policy.

`--allow-all-tools`

Grants the AI model access to all available Copilot CLI tools during code generation. Remove this flag to restrict tool access.

Note: This topic focuses on configuring AI-assisted pull request reviews with Code Genie. For a complete overview of Code Genie capabilities, see Code Genie on page 199.

3. Commit and push the updated `code-config.jsonc` to the main branch.
4. Validate the end-to-end **Code Genie** workflow by creating a test branch and opening a pull request to the main branch.

../videos/code-genie-ai-review.mp4

a. Create a test branch in the Control repository, make a minor change, commit and push the changes, and open a pull request targeting main.
b. Confirm that Code Genie shows a yellow dot next to its name, indicating the `Copilot` container is starting.
c. Confirm that Code Genie adds it comments to the pull request after the Copilot review completes and that the merge button becomes active.

5. Verify the webhook on page 212 that was automatically configured during **Loop** creation.
6. Verify branch protection rules on page 233 and Code Genie reviewer settings that were auto-configured during Loop creation.

Results

You have configured AI-integrated Code Genie on your Control repository. Every pull request to the main branch now triggers an interactive Copilot AI code review session. Code Genie posts AI-generated code suggestions as pull request comments for developer review.

What to do next

After completing this configuration, you can:

- Switch to a different AI model by updating the `--model` argument in the Copilot container command in Code Genie's `code-config.jsonc`.
- Enable the IBM Bob Shell container as an alternative AI reviewer by uncommenting the `Bob` dev container block in `code-config.jsonc`.
- Add further dev container blocks to run additional capabilities such as unit testing or static code analysis alongside Copilot. See Code Genie on page 199.

Trivy scanning with Code Genie

A Control repository that is created as part of creating a HCL DevOps Loop will be automatically configured to perform security scanning with **Trivy**. This ensures that every pull request is automatically scanned for problems such as vulnerabilities, security breaches, misconfigurations, etc. before code reaches the main branch. Use this procedure when you want to enforce severity-based merge protection across your team.

Before you begin

Before you begin, ensure that you have:

- Administrator access to your team space.
- Created a teamspace. See Creating a teamspace on page 132.
- Created a **DevOps Loop** so that the Control repository, webhook, branch protection, and default `.devops-loop/code-config.jsonc` are provisioned automatically. See Creating a loop on page 140.
- Read the Code genie on page 199 topic to understand how Code genie works with features like Trivy.
- Familiarized yourself with the Trivy CLI reference for available scan parameters.
- Ensured your team follows a branching strategy on page 233with a protected main branch.

Note: If your security policy doesn't require running Trivy scans on every pull request, you can disable them. For more information, see Disabling Trivy Scanning and Code Review on Pull Requests on page 206.

About this task

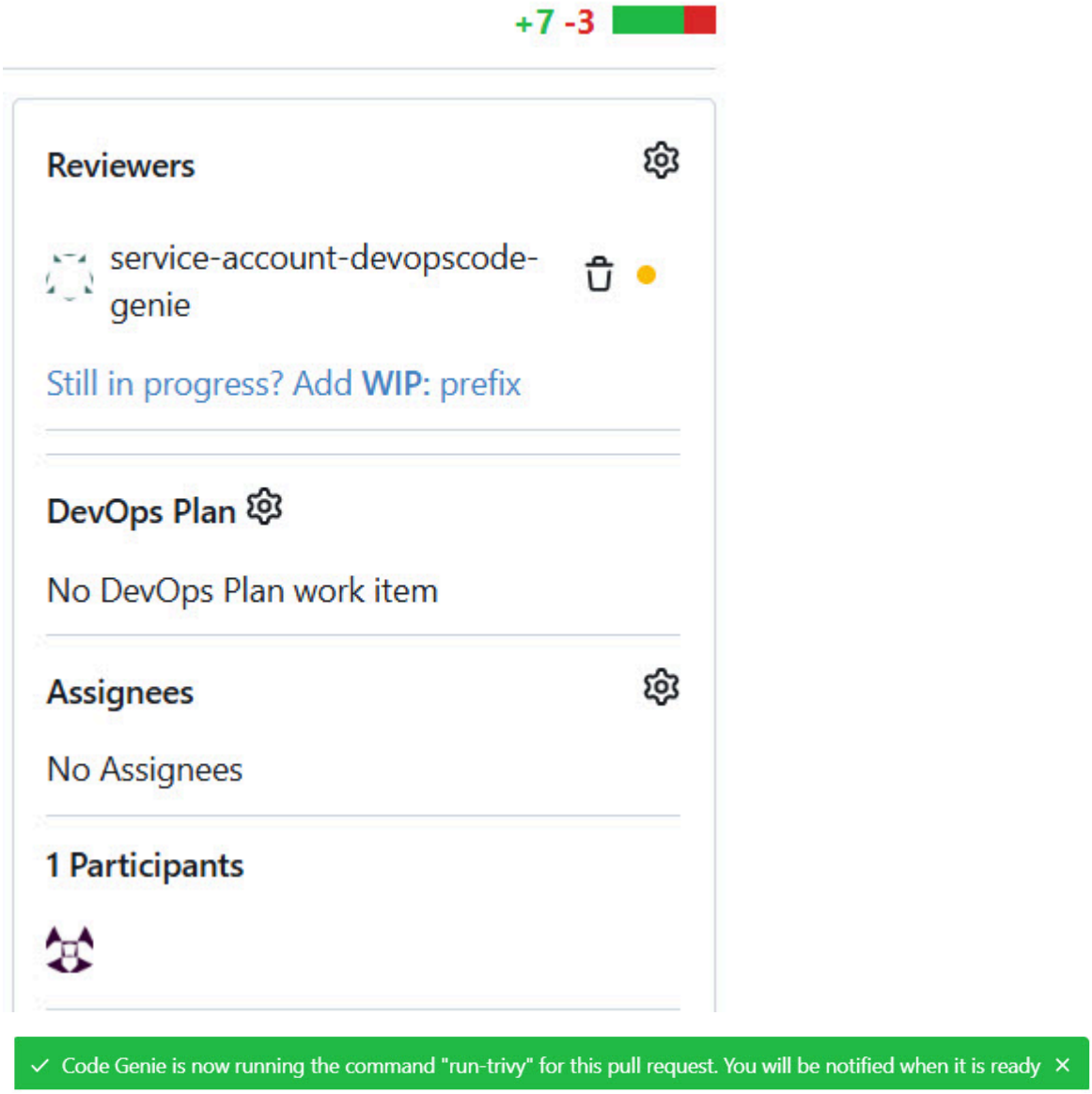
Trivy scans every pull request for vulnerabilities (CVEs), IaC misconfigurations, leaked secrets, and license violations. Code Genie (`service-account-devopscode-genie`) posts the scan results as pull request comments and blocks merges when findings equal or exceed the configured rejection threshold. You can customize the scan behavior in `.devops-loop/code-config.jsonc` by setting the rejection threshold (`--reject`), the scanner types (`--scanners`), and types of severity issues to scan for.

Important: Automatic configuration of the webhook, branch protection rules, and `.devops-loop/code-config.jsonc` only applies to Control repositories created through the Loop creation process. If you create a Control repository manually at a later stage, you must manually add the webhook, configure branch protection rules, and create the `.devops-loop/code-config.jsonc` file for Trivy scanning to work.

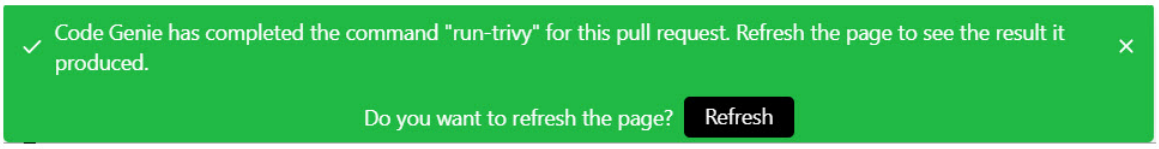
Important: When a pull request is closed, the `TrivyScan` dev container associated with it is automatically terminated. Any other open pull requests waiting for Code Genie scan results at that point may not receive their comments because the container is no longer running.

These are temporary limitations in the current release.

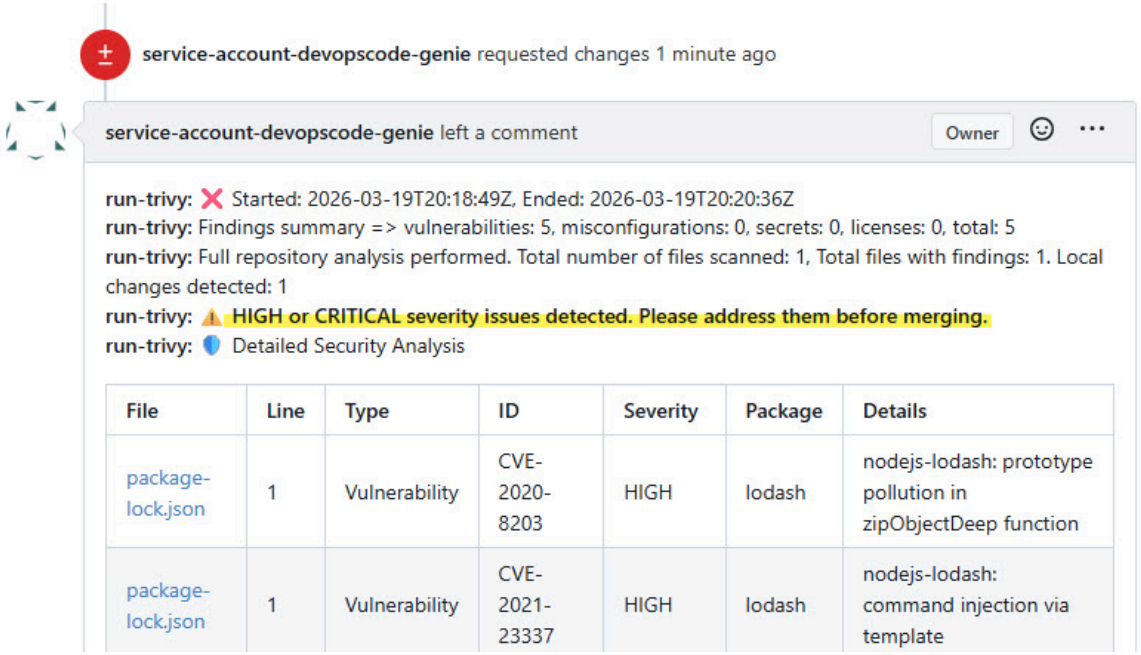
1. Validate the end-to-end **Trivy** workflow by creating a test branch with a vulnerable dependency and opening a pull request to the main branch.
    a. Create a test branch in the DevOps Control repository, add a dependency with a known vulnerability, commit and push the changes, and open a pull request targeting main.
    b. Confirm that Code Genie shows a yellow dot next to its name, indicating the scan is running, and that the merge button is disabled.



c. Wait until a pop-up appears that tells Trivy scanning is completed. It usually takes around 20 seconds, but it depends on the size of the Control repository.



d. Review the Code Genie pull request comments and confirm that HIGH or CRITICAL findings are blocking the merge, if the `--reject` is set to high.



service-account-devopscode-genie requested changes 1 minute ago

service-account-devopscode-genie left a comment — Owner

run-trivy: ❌ Started: 2026-03-19T20:18:49Z, Ended: 2026-03-19T20:20:36Z
run-trivy: Findings summary => vulnerabilities: 5, misconfigurations: 0, secrets: 0, licenses: 0, total: 5
run-trivy: Full repository analysis performed. Total number of files scanned: 1, Total files with findings: 1. Local changes detected: 1
run-trivy: ⚠️ **HIGH or CRITICAL severity issues detected. Please address them before merging.**
run-trivy: 🛡️ Detailed Security Analysis

File	Line	Type	ID	Severity	Package	Details
package-lock.json	1	Vulnerability	CVE-2020-8203	HIGH	lodash	nodejs-lodash: prototype pollution in zipObjectDeep function
package-lock.json	1	Vulnerability	CVE-2021-23337	HIGH	lodash	nodejs-lodash: command injection via template

e. Address the security issues, push the fix, and allow Code Genie to re-run the scan.

f. Confirm that Code Genie approves the pull request after the clean scan and that the merge button becomes active.

service-account-devopscode-genie left a comment — Owner

run-trivy: ✅ Started: 2026-03-19T20:10:11Z, Ended: 2026-03-19T20:10:14Z
run-trivy: Findings summary => vulnerabilities: 0, misconfigurations: 0, secrets: 0, licenses: 0, total: 0
run-trivy: Full repository analysis performed. Total number of files scanned: 1, Total files with findings: 0. Local changes detected: 1
run-trivy: ✅ Security scan passed (No issues at HIGH or CRITICAL).
run-trivy: ✅ No security issues found.

✓ This pull request can be merged automatically.
There is no key available to sign this commit.

Create merge commit

▶ View command line instructions

Write Preview

2. Verify the webhook that was automatically configured during **Loop** creation.

a. Go to **Settings > Webhooks** in the repository.

b. Confirm that the webhook shows the following values.

- Target URL: `http://devops-code-container-mgr:3005/api/pull-request`
- HTTP method: POST

- Content type: `application/json`
- Trigger events: Pull Request as well as PR Synchronized
- Status: Active

Warning: Do not delete or modify the auto-generated webhook. Doing so breaks Trivy scanning.

3. Open `.devops-loop/code-config.jsonc` in the **Control** repository and customize the Trivy scan parameters to match your security policy.

   The default configuration is as follows:

```
{
  "code-genie": {
    "pull-request": [
      {
        "dev-container": "TrivyScan",
        "interactive": false,
        "custom-commands": [
          {
            "run-trivy": "cp /opt/run-trivy.sh /usr/code/ && chmod +x /usr/code/run-trivy.sh &&
 REPO_URL=https://${platform-fqdm}/control/${repo-owner}/${repo}/src/branch/${pr-branch}/ /
usr/code/run-trivy.sh ${code-folder} ${result-json} --reject=high --trivyArgs=\"--scanners
 vuln,misconfig,secret,license --license-full --severity UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL\""
          }
        ]
      }
    ]
  }
}
```

   You can adjust the following parameters:

   - `--reject` — sets the minimum severity level that blocks a pull request merge. Accepted values are `never`, `low`, `medium`, `high`, and `critical`. Also, `never` means that the PR will never be rejected, and that only Trivy scan results will be provided as PR comments.
   - `--scanners` — sets the scanner types to run.
   - `--severity` — sets the severity levels that Trivy reports on.

4. Verify branch protection rules on page 233 and Code Genie reviewer (auto-configured during Loop creation).
5. Commit and push the updated `code-config.jsonc` to the main branch.

Results

You have configured Trivy security scanning on your Control repository. Every pull request to the main branch now triggers an automatic Trivy scan. Code Genie approves or blocks the merge based on the `--reject` threshold you defined in `code-config.jsonc`.

What to do next

After completing this configuration, you can:

- Adjust the `--reject` threshold or `--scanners` list in `code-config.jsonc` as your security policy evolves.
- Add further dev container blocks to run additional capabilities such as unit testing or static code analysis alongside Trivy. See Code Genie on page 199.
- Review the Trivy CLI reference for the full list of available scan arguments.

Branching strategy with a protected main branch

Verify that branch protection rules and Code Genie are correctly configured on your Control repository to enforce a pull request-based workflow on the main branch.

1. Navigate to **Settings > Branches > Branch Protection** in the **Control** repository.
   ../videos/branch-protection.mp4
2. Verify branch protection rules and Code Genie on page 199 reviewer (auto-configured during Loop creation).

    a. Confirm that the **main** branch protection shows the following:
        - Direct pushes to main are blocked.
        - Require pull request reviews before merging: ENABLED. At least one PR approval is needed.
        - Required reviewers list contains: `service-account-devopscode-genie`.
        - Block merge on rejected reviews: ENABLED.

Managing Git authentication for developers

In this topic, you will learn how to manage authentication when using Git on your system to access Control repositories within DevOps Loop deployment.

There are three types of credentials you can use with Control in a Loop deployment:

- **SSH keys**: For information, see the Using an SSH key (for Desktop) section in the Control Documentation.
- **Personal access token via HTTPS**: For information, see the Using HTTPS (not with OAuth2 support) section in the Control Documentation.
- **OAuth credentials with Git Credential Manager via HTTPS**: To configure your Git client in the development environment for HTTPS communications with Control in a DevOps Loop deployment, add settings similar to the following example using the Git command line. Replace *`yourhost.example.com`* with your Loop server's hostname:

```
git config --global credential.https://yourhost.example.com.provider generic
git config --global credential.https://yourhost.example.com.oauthClientId git-credential-manager
git config --global credential.https://yourhost.example.com.oauthTokenEndpoint /auth/realms/devops-
automation/protocol/openid-connect/token
git config --global credential.https://yourhost.example.com.oauthDeviceEndpoint /auth/realms/devops-
automation/protocol/openid-connect/auth/device
git config --global credential.https://yourhost.example.com.oauthAuthorizeEndpoint /auth/realms/devops-
automation/protocol/openid-connect/auth
git config --global credential.https://yourhost.example.com.oauthAuthModes DeviceCode
git config --global credential.https://yourhost.example.com.oauthScopes api:control:all
```

This configuration ensures secure and efficient access to your repositories.

Note: After any Loop upgrade, you should remove cached credentials from the previous use of Git Credential Manager. Refer to the documentation for your Git Credential Manager to find instructions on how to clear the cached credentials.

Build

Build offers the capability to build applications, manage continuous integration workflows, and store build artifacts.

Build uses a template-driven approach, enabling the creation of projects based on predefined templates that can be extended with additional process and workflow templates. These templates help standardize build procedures, enforce source control policies, integrate testing practices, and manage reporting workflows.

System requirements

Build runs on Windows and Unix-based systems. While the minimum requirements provided below are sufficient for an evaluation, use server-class machines for production deployments.

For more requirement details, you can dynamically generate a system requirements report using the Software Product Compatibility Reports (SPCR) tool.

Server installation for best results

- **Two server-class computers**

  Use two computers for the server: a primary server and a standby for fail-over. In addition, host the database on a separate computer in a fail-over configuration.

- **Database servers:** Separate computers.
- **Processor:** 2 processors and at least 2 cores for each.
- **Memory:** 8 GB
- **Java:** 32-bit or 64-bit JDK.
- **Disk Space:** Individual requirements depend on usage, retention policies, and application types. In general, the larger the number of retained artifacts in the Build artifact repository CodeStation, the more storage is required.

  Note:

  CodeStation is installed when the Build server is installed.

  For production environments, use the following guidelines to determine storage requirements:

- 10-20 GB of database storage is typically sufficient for most environments.
- Use this format to calculate CodeStation storage requirements:

```
Average artifact size x Number of versions imported per day x Average number of days before cleanup
```

- Approximately 1MB of database storage per build. This value varies according to the build process.

For further assistance in determining storage requirements, contact IBM Software Support.

- **Network:** Gigabit (1000) Ethernet with low-latency to the database.

Minimum agent requirements

Designed to be minimally intrusive (typically, an idle agent uses 5MHz of processor power), agents require 64-256 MB of memory and 100 MB of disk space. Agents can run with a 32-bit or 64-bit JDK. Additional requirements are determined by the processes that the agent runs. In production environments, install agents on separate computers.

Minimum server installation requirements

- **Operating System:** Windows 2000 Server (SP4) or later, Linux (any distribution). Ask IBM Software Support about other operating systems.
- **Processor:** Single core, 1.5 GHz or faster.
- **Memory:** 2 GB, with 512 MB available to Build.
- **Java:**

DevOps Build version	Java version
7.1.0.2	JRE 17
7.1.0.1	JRE 11
7.1.0.0	JRE 11 or JRE 8

- **Disk Space:** 300 MB minimum. More is required when you start publishing build artifacts and using the embedded database.

Agent Minimum Requirements

Agents require 64-256 MB of memory and 100 MB of disk space. Agents can run with a 32-bit or 64-bit JDK. Additional requirements are determined by the processes that the agent runs. In production environments, install agents on separate computers. For requirement information, generated a report using the Software Product Compatibility Reports (SPCR) tool.

Overview of Build

Build is a continuous integration and build-management system.

Build is a distributed, multi-platform, enterprise-scale build management tool. Build uses a template-driven system to configure and run builds. Projects are created from project templates and are configured by assigning other templates to them, such as process and workflow templates. The templates can define standard build processes, source control rules, and testing and reporting processes.

With the flexible role- and team-based security system, you can create an environment that conforms to your organization. For example, you might maintain a design team that creates templates for other teams. Development teams can create projects quickly by configuring the preset templates. Values for template properties can be supplied at design time or at run time.

Every time that you build a project, build artifacts are added to an embedded artifact and dependency management system, CodeStation. CodeStation helps manage complex projects that can depend on numerous sub projects and libraries. Because CodeStation is part of Build, third-party management systems are not necessary. Builds are repeatable, standardized, and traceable.

Distributed agents do the actual work of build processing. An agent is a lightweight process that runs on a host computer and communicates with the server. By handling the build activities, agents distribute the load of concurrently running builds and provide enterprise-level scalability.

The management tools in Build provide project status at any point in the build cycle, and simplify capturing project data. Enterprise-level metrics are available for these areas:

- Compliance
- Governance
- Data aggregation and reporting
- Auditing and logging
- Version and change control

Concepts

To better understand Build, familiarize yourself with these concepts.

Projects

Projects in Build represent the components and applications that your company develops. Each project has its own pipeline for release or distribution. Projects often correspond to how development is organized in your source control management system. Projects might also be composites of other projects that use a dependency management system. If components are developed independently and then composed into applications, the components themselves can become projects that support other projects that represent applications or products. Projects are created with project templates.

Build processes

A build process is a configuration that you run to conduct a build on some source material.

The build processes define:

- Everything that happens during builds and everything that is required by them.
- Where to get source code and dependencies.
- What tools are required to build.
- What tests need to run.

The build processes are created with process templates which typically produce artifacts and are attached to projects, which can have any number of build processes attached to them.

Source configurations

A source configuration identifies the source material for a build. They can define the branch or label to pull source material from, or use other criteria. Source configurations are attached to build processes. A build process can have multiple source configurations assigned to it if it pulls source code from multiple locations. Source code locations can be in multiple repositories and from any type of source control system. Source configurations are created with source configuration templates.

Source repositories

A source repository configuration supplies the information that is required to connect to an external source repository. Because source repository configurations do not define the details of any particular source, they are highly reusable.

Dependencies

Dependencies are configured to provide artifacts and build triggers between projects and third-party artifacts, such as open source libraries. When you create a dependency for a build process, you identify the artifacts that your build requires to run or build. Dependencies are highly configurable; you can specify criteria for version, or status, or other characteristics. Dependencies can trigger builds, which can work bi-directionally by pushing or pulling. Pull dependencies retrieve material from other sources, and push dependencies send material to other targets. If you have a common component project, you might trigger builds when the common project changes.

Templates

Templates are reusable configurations that define projects and processes. Each template encapsulates a type of project, such as a source configuration, workflow, job, or process. You can create templates properties and define their values at design time or run time. For instance, a template might always call a particular type of build script, but the name or location of the script might vary by project. Templates provide a high level of reuse because you decide the number of properties and when their values are defined.

Processes

A process defines the jobs and steps that go into a build. Processes are created with process definition templates, and like all templates, they are reusable. A number of built-in steps are available, but most of steps come from plug-ins. Steps are collected into jobs and jobs are composed into processes. When you place a job in a process, you specify several options, including the agent pool, the working directory, and others. With processes, you can run jobs in parallel and then join them

later. Jobs can also be iterated to run a number of times. Each job run can have its own properties, which are useful for multi-platform, parallel builds.

Build lives

A build life is record of a build. It records the inputs, the build activities, the produced artifacts, and activities that are done after the build. A build life tracks the source, the dependencies, the computer that is used, and the deployment or release of the produced artifacts. All completed steps and their output are recorded. Tests can be recorded, along with defects, features, and stories. Each build has a unique build life.

Secondary processes

A build process produces a new build; a secondary process runs on a build life. Secondary processes can be run only on an existing build life with at least one completed build process. Typically, secondary processes are used for tests or analytic tools. If one of your test suites takes an unusually long time, you might want to schedule it to run nightly in a secondary process.

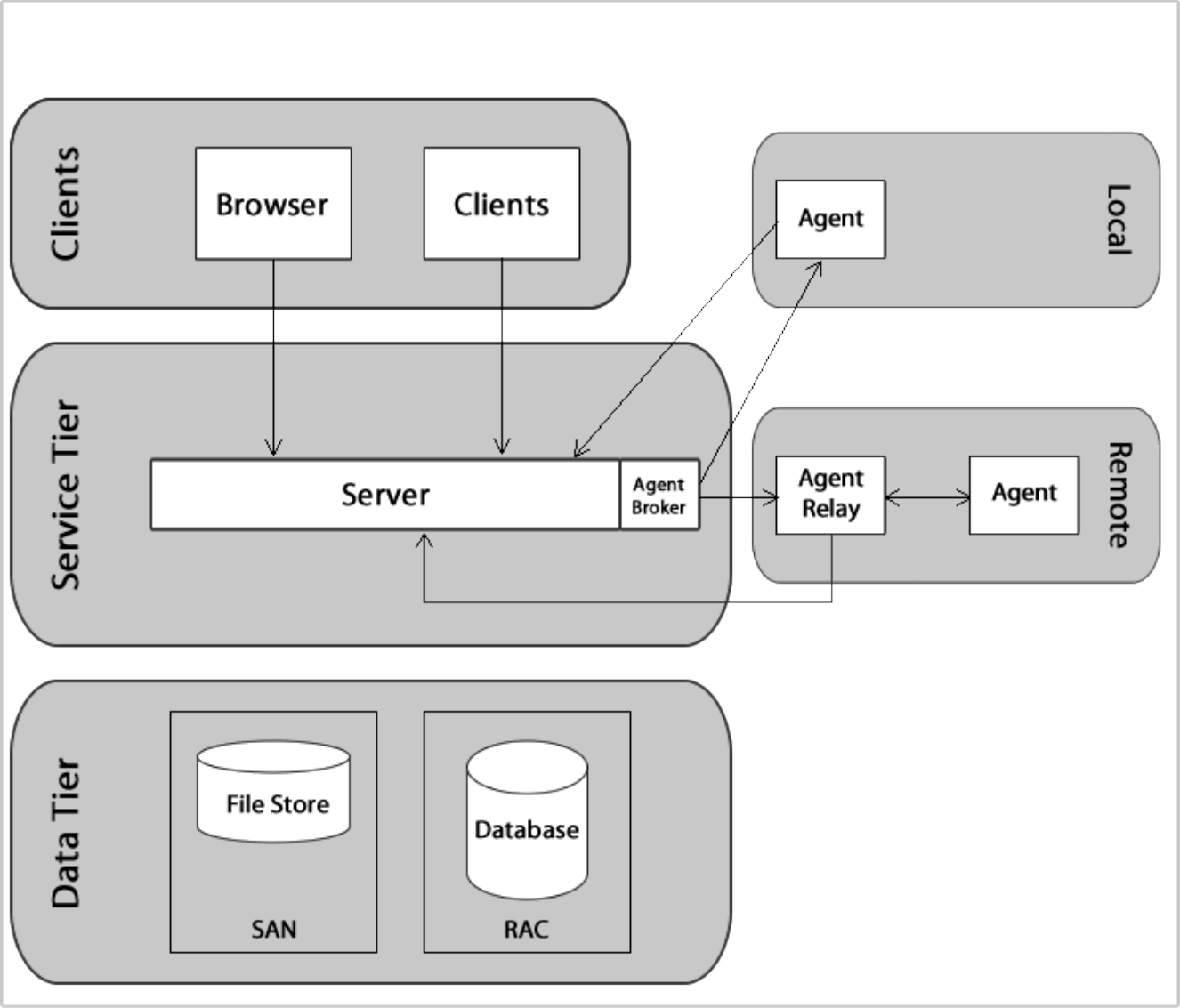
Architecture

Overview of the Architecture

The Build architecture consists of a centralized server and distributed agents. The server provides a data store, user interface, and coordinates agents among other things. All build activities occur on agents, and agents act only when the server tells them to.

The Build server consists of a service tier and a data tier. The service tier consists of a central server that provides a web server front-end and core services, such as workflow, agent management, security, and others. A service can be thought of as a self-contained mechanism for hosting a piece of processing logic. The server orchestrates builds with agents that are distributed throughout the network. The server exposes several interfaces. The first is a web user interface that is exposed through HTTP or HTTPS and is primarily used by users with browsers. Web services use HTTP or HTTPS and are primarily used by agents for some types of communications, and for integrations with external tools. The last interface is used for server-agent communication and is done with JMS (discussed later) and uses TCP/IP.

Architectural overview



JMS server-agent communication and associated web services are stateless forms of communication. "Stateless," as used here, means that the server retains little session information between requests and each request contains all the information that is required to handle it. With this setup, the communication system performs well without a lot of resource overhead.

Server-agent communication is used to send work to agents and the results back to the server. Because JMS connections are persistent and are not based on a request-response protocol, Build does not have to continually open and close connections. With persistent connections, the server can communicate with agents at any time while it remains secure and scalable. The server listens on a port for agents to connect to it. For added security, agents do not listen on ports and make connections to the server only to provide two-way communication. This arrangement simplifies firewall configuration because all connections are initiated in one direction, from the agent to the server.

Many services in Build are exposed as REST-type (representational state transfer) web services. These services are used to send and receive information over HTTP or HTTPS. The information can be formatted in XML or JavaScript Object Notation (JSON). The REST-style services are stateless and ensure that requests include all the data that the server requires to respond coherently.

The data tier stores information in a relational database and on the file system. The data tier's relational database stores configuration and run time data. The data tier's file store contains log files and build artifacts. Artifacts are stored in a system that is called CodeStation, which is also a dependency management system. CodeStation stores the build artifacts in a content-addressable storage format that minimizes storage requirement by not storing the same files multiple times. By default, file storage is contained in the `var` directory of the server installation. The location of this storage can be changed with the server configuration, or you can map the directory to a network addressable storage location.

Service tier

The Build server provides various services, such as: the user interface, process engine, and security services, among others. The REST-based user interface provides the web-based front end that you use to create projects and define workflows; request processes, and manage security and resources, among other things.

When a workflow is requested, many services are used to fulfill the request. Workflow requests are initiated with the user interface or a REST service.

Table 6. Services

Service	Description
User Interface	Used to manage the configuration of all items and see the current and historical runtime information of the system.
Process Engine	Does processing for builds and secondary processes. This service also uses the agent manager service to acquire agents for jobs and run steps on agents. This service also sends events about job starts and completions that are used by notification and dependency services for sending notification and triggering other jobs.
CodeStation	Is a dependency management system and artifact repository. CodeStation can store and process dependency configurations, with which builds can retrieve the artifacts that they require at build-time. This resolution is also stored with the build life as a permanent history of what was used. CodeStation also provides an interface as a Maven repository.
Agent manager	Tracks agents and their offline and online status. This service also controls access to agents if a workflow must run a step on an agent. Agent manager also stores the properties of an agent and can be used to upgrade or restart an agent remotely.
Plug-in manager	Build can interact with virtually any system through its extensible plug-in system. Plug-ins provide functions by breaking-down tool features into automated steps. Plug-ins can be configured at design- and run time. When a plug-in step runs, the controlling agent calls its runtime process to run the step.

Table 6. Services (continued)

Service	Description
Event service	A message bus with publishers and subscribers that provides messaging between services in an asynchronous method.
Notification manager	Notifies users about events by using notification schemes, which consist of the when, who, what, and how of notifications. When is the event that triggers a notification. The who is to whom the notification is sent. What is the content that is sent which, is generated from templates. How is the notification channel and can be email or instant message.
Security	Includes authentication and authorization functions. The authentication is configuration for determining who a user is and might use an internal database, LDAP, Active Directory, or a Single Sign-On service. The authorization is configuration for what a user can do. The security service includes tools for assigning users to groups, and for assigning users and groups to roles.

Data tier

The Build data tier contains a relational database and a file storage system.

Relational databases

Build supports Oracle, Microsoft™ SQL Server, and MySQL relational databases. The provided Derby database, though sufficient for proof-of-concept work, is insufficient for the enterprise. Ideally, the database you use is configured for high-performance and is backed up regularly.

Database storage of 10-20 GB is typically sufficient for most environments. The following configurations are sufficient in most cases:

- Oracle RAC
- Microsoft™ SQL Server, clustered
- MySQL, cluster

File storage: CodeStation

The data tier also provides log file and CodeStation artifact storage.

Artifacts represent items such as files, images, databases, configuration materials, or anything else that is associated with a software project. By default, these artifacts are stored in the `var` subdirectory in the Build server installation directory. In an enterprise environment, the default installation might not be ideal. See CodeStation relocation on page 242 for enterprise options.

The artifact repository uses content addressable storage to maximize efficiency while it minimizes disk use. The repository tracks file versions and maintains a complete history for all artifacts. Maximizing efficiency is important because the repository stores files that are much larger than source files.

The artifact repository comes with a client application that provides remote access to the repository. Using the client application, you can add or modify files, create packages, retrieve files, and view the history of changes. The client application provides a file transfer capability that can be used to deliver files to target servers during deployments. This built-in transfer mechanism verifies the integrity of all transferred files against their expected cryptographic signatures. This verification ensures that files were not corrupted during transmission or tampered with during storage. In addition to the client application, the artifact repository provides REST-based web services.

CodeStation relocation

By default, log files and CodeStation artifacts are stored in the `var` subdirectory within the Build server installation. You can store this data in another location. Ideally, the new location is on a network storage system that is regularly synchronized with an off-site disaster recovery facility. In addition, make sure that the Build server has a fast network connection to the storage. Agents do not need access to the storage location. In UNIX environments, you can use symbolic links from within the `var` subdirectory to network storage. On Windows platforms, there are several options for redirecting logs and artifacts, including mklink, which is supported in Windows 7 and later.

You can also move file storage to a location other than the `var` directory by setting a `server/var.dir` property for the Build server. To set the property, edit the server's `set_env` file in the server's `bin` subdirectory. Edit the definition of `JAVA_OPTS` to include the redirection property, as shown in this example:

```
-Dserver/var.dir=/alternate/var/path
```

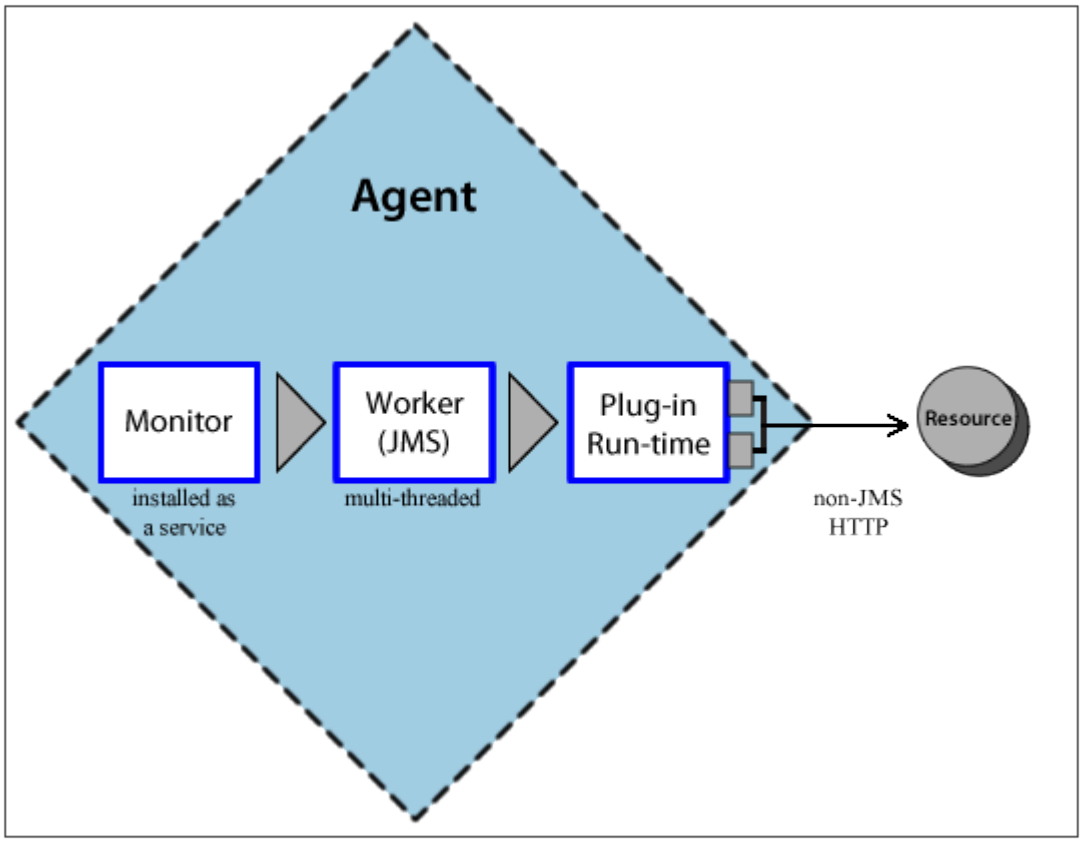
Distributed teams can also take advantage of Build location-specific CodeStation proxies to improve performance and lower WAN use.

Agents

An agent is a lightweight process that runs on a host computer and communicates with the Build server. Agents are the workers of the Build architecture. Agents complete the actual build activity, which distributes the load of concurrently running builds and provide scalability. After an agent is started, the agent opens a connection to the Build server. Communication between server and agents uses a JMS-based (Java Message Service) protocol and SSL, which can optionally enforce mutual authentication. This communication protocol is stateless and resilient to network outages.

Although an agent is characterized as a single process, technically an agent consists of a monitor process and a worker process. The monitor is a service that manages the worker process. It is responsible for starting and stopping the worker, handling restarts, and completing upgrades. If the worker fails for an unexpected reason, the monitor restarts it. The worker is a multi-threaded process that connects to the server and completes any work that the server sends to it. Agents do not have to be upgraded every time that the server is upgraded. The server maintains a minimum agent version property, which is the minimum version that is required to work with the server. When agents are updated, it is to take advantage of new features or fix bugs. Agents can be upgraded with the installation process on the host computer or remotely through the server web user interface. After an agent is installed and connected to the server, it is displayed in the web user interface where it can be activated and placed into agent pools. A newly installed agent does not receive work until you activate it.

Agent processes



Agents are an important part of Build scalability. By adding more agents, the throughput and capacity of the system increases and can scale to serve even the largest enterprise.

Server-agent communication

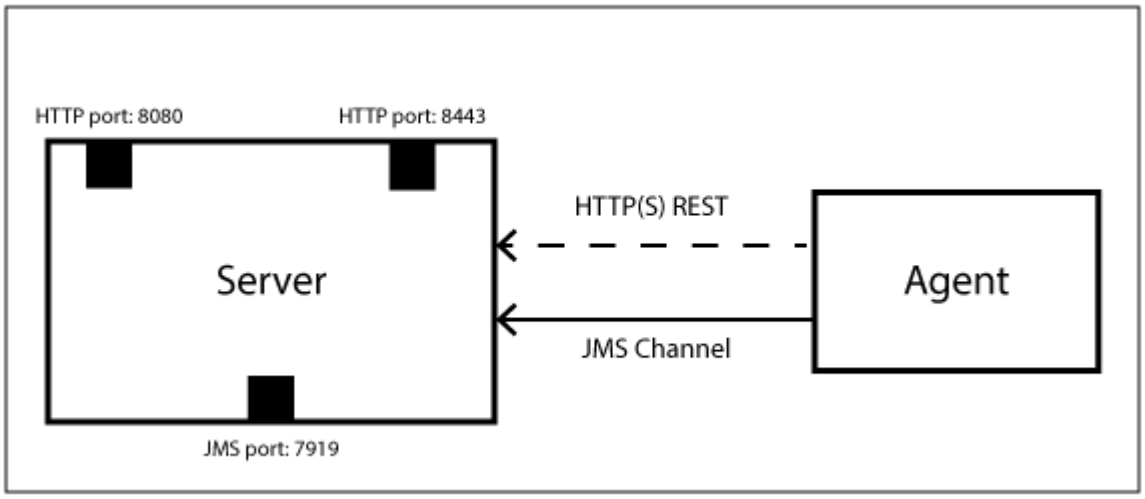
Most server-agent communication is done with JMS, but some agent activities such as uploading logs, test results, or artifact files to CodeStation use HTTP or HTTPS as needed. The JMS channel is the primary control channel for Build. It is the channel that the server uses to send agent commands. By default, the server listens on port 7919 for JMS, port 8080 for HTTP, and port 8443 for HTTPS. These ports are configured during the server installation. The agent needs to know only the server's host and JMS port. The web interface URL is sent to the agent when it is needed.

The server's agent manager service uses JMS for all server communications and for sending commands, such as "run step," to the worker process. The worker process uses JMS for system communications, and HTTP REST services when it processes plug-in steps or retrieves information from the server.

Stateless server-agent communication provides significant benefits to performance, security, availability, and disaster recovery. Because each agent request is self-contained, a transaction consists of independent message, which can be synchronized to auxiliary storage as it occurs. The server or agent can be taken down and brought back up without repercussion (other than lost time). If communications fail mid-transaction, no messages are lost. When reconnected, the server and agent automatically determine which messages got through and what work was completed. After a connection

outage, the system synchronizes the endpoints and recovers affected processes. The results of any work that an agent completes during the outage are communicated to the server.

Agent-server communication



In the accompanying figure, the arrow represents the direction in which communications was established, but the flow can be in both directions with JMS.

Mutual authentication

With Secure Socket Layer (SSL) technology, clients and servers can communicate securely by encrypting all communications. Data is encrypted before it is sent and decrypted by the recipient. Communications cannot be deciphered or modified by third-parties. SSL can be used in two modes:

In unauthenticated mode, communication is encrypted and decrypted but endpoints do not have to verify the other's credentials. By default Build uses this mode for its JMS-based server-agent communication.

In mutual authentication mode, communications are encrypted and decrypted, but endpoints are also required to authenticate themselves by providing certificates. A *certificate* is a cryptographically signed document that is intended to assure to others the identity of the endpoint that uses the certificate. Mutual authentication is enabled on a per-endpoint basis.

When mutual authentication mode is enabled on an endpoint, the endpoint requires that the other endpoint that it is communicating with has a trusted certificate. A certificate is trusted if that certificate is signed by another trusted certificate such as a certificate authority or is a trusted certificate itself. A certificate is trusted if it is imported into the endpoint's keystore as a trusted certificate.

When the server has mutual authentication enabled, the agent certificates that are used must be signed by a trusted certificate authority or be imported into the server's keystore. The same applies for the agent. The server and the agent are not required to have mutual authentication enabled together. The setting applies only to the endpoint and how it accepts certificates of endpoints it communicates with.

Tutorials

This section contains tutorials for users of all experience levels. If you are new to Build, this tutorials introduces the most-used product features in an easy-to-follow way. You do not need additional resources to complete the tutorial.

Create a Maven build project

In this tutorial, you create a project that builds the open source product JUnit. The source code is downloaded from GitHub (an open source code repository) and built with Maven.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a Git repository from which your project downloads the source for JUnit.
- Create a build process and associate an agent pool to it.
- Create a job that checks-out the JUnit source from GitHub and uses Maven to build JUnit, and run tests and generate test reports.
- Create templates that define properties. You supply values for these properties when you create the project.
- Create a project that is based on the templates you created.
- Run a build and monitor its progress.
- Examine build details and troubleshoot problems.

Time required

2 hours

Introduction

In this tutorial, you create a project that builds the open source product JUnit. The source code is downloaded from GitHub (an open source code repository) and built with Maven.

The project that you create in this tutorial introduces you to the most-used product features. The workflow that you create contains the job types you use everyday.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a Git repository from which your project downloads the source for JUnit.
- Create a build process and associate an agent pool to it.
- Create a job that checks-out the JUnit source from GitHub and uses Maven to build JUnit, and run tests and generate test reports.
- Create templates that define properties. You supply values for these properties when you create the project.
- Create a project that is based on the templates you created.
- Run a build and monitor its progress.
- Examine build details and troubleshoot problems.

Time required

This tutorial takes approximately 2 hours to finish. If you explore other concepts that are related to this tutorial, it might take longer to complete.

System requirements

To run this tutorial, you need the following resources:

- An installation of Build, version 6.1 or later.
- An Build agent installed.
- An installation of Maven on the computer where the agent is installed.

Prerequisites

To complete this tutorial, you must have an account on the server with permissions to create and edit projects, jobs, and repositories, and to create and run processes.

Configure the tutorial system

To run the tutorial project, you must configure an *agent*, install the required plug-ins, and create an artifact set.

Before you begin

Before you begin the tutorial, make sure that you completed the Prerequisites on page 246.

About this task

To complete the tutorial, you must have an agent that is running on a target computer. An agent is a lightweight process that runs on a target computer and does the actual work of building the project. This target computer can be a virtual image, or a physical computer, including the one where the server is installed. You must have access to this computer and be able to install the agent on it.

To prepare the tutorial system:

1. Prepare a computer for use as the target system.
2. Install the plug-ins used in the tutorial.
   The tutorial uses the following plug-ins: Git, Maven, and JUnit. The Maven plug-in is installed with the product. Download and install the Git and JUnit plug-ins. Plug-ins can be downloaded from the Build plug-in site: Build plug-ins website. For information about installing plug-ins, see Installing plug-ins on page 351.
3. For performance reasons, agents are generally not installed on the same computer where the Build server is located. For the tutorial, installing the agent on the computer where the server is located is fine.
4. Configure the agent on page 342.
   Your agent must be included in the list of configured agents on the **Agents** page with a status of `Online`.
5. Create an agent pool and add the installed agent to it. on page 345
6. Install the open source tools Git and Maven on the computer where the Build agent is located.
   Add the home directory for each tool to the *PATH* variable on the target system. Add the *MAVEN_HOME* variable to the *PATH* variable.

7. Ensure that the artifact set included with the product, **maven,** is available. If it is not installed, create the artifact set by completing these steps:
    a. From the Build dashboard, click **System**, then click **Artifact Sets**.
    b. In the **Artifact Sets** page, click **Create New Artifact Set**.
    c. In the **New Artifact Set** dialog box, in the **Name** field, type `maven`.
    d. Click **Save**.
8. **Optional:** Configure Java™.
   The agent that runs the tutorial project must be configured to use a Java™ JDK instance, and not a JRE. If the agent is configured to use a JRE (identified by the agent's `java.home` property), reconfigure the agent to point to a JDK. Alternately, you can add the `JDK_HOME` property to the `installed.properties` file, which is in the *`agent_installation`*`/conf/agent` directory. In addition, the tutorial's Maven Build step must reference that property with the `JAVA_HOME` setting.

Results

Your Build environment is ready to being the tutorial.

The tutorial project is also available in a file that you can download and import. To import the tutorial project, follow these steps:

1. Navigate your browser to the file repository URL:http://download.boulder.ibm.com/ibmdl/pub/software/rationalsdp/documentation/product_doc/UrbanCode/uBuild/.
2. Download the file named `DemoProject1.xml`.
3. On the Build **Project** page, click **Import**, and then select the downloaded file.

   Note: Make sure that you install the plug-ins described earlier in this topic before you import the file.

4. In the **Import** window, click **Import**. The server imports the project. The server does not import plug-ins when it imports a project, so you still need to install the plug-ins described earlier in this topic.

Lesson checkpoint

The server and agent are configured for the tutorial, and the files that represent artifacts are ready to use. If you installed the agent on a remote system, you must know the credentials that are required to access the remote system.

Lesson 1: Create a Git repository

Repository integrations enable Build to checkout code, access change logs, and do other project-related activities.

Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the tutorial system on page 246.

About this task

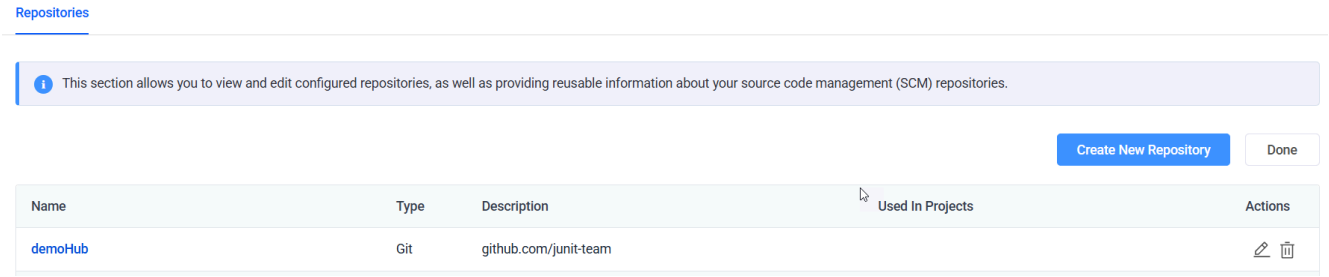
1. On the left navigational panel, click **System > Repositories**.
2. Using the **Repositories** page, click **Create New Repository**.
3. In the **Source Plug-in** list, select **Git** and then click **Set**.
4. In the **Name** field, type `demoHub`.
5. In the **Repository Base URL** field, type `https://github.com/junit-team`.
   This GitHub project contains the JUnit source code.
6. Click **Save**.

Lesson checkpoint

In this lesson, you created a repository based on the Git integration.

About this task

The DemoHub repository is listed on the **Repositories** page.



For more information about repositories, see Repositories on page 330.

Lesson 2: Creating a job and adding steps

A job is a series of steps that, taken together, complete some activity, such as building a project. The steps are run one at a time in the order you specify.

Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the tutorial system on page 246.

About this task

In this lesson, you create a job and add several steps to it.

Creating a job

Most job steps are created from plug-ins. Plug-in steps can be thought of as distinct pieces of automation.

About this task

By combining steps from various plug-ins, you can create fully automated build processes. The number of steps that are provided by a particular plug-in can vary, and each step has a variable number of properties. Property values can be supplied when you define a step or at process run time.

Note: Several plug-ins are included in the product, and you can download other plug-ins as described earlier in the tutorial on page 246. Some steps, such as Assign Status and Create Stamp, are not derived from plug-ins, but you use them in the same way you use plug-in steps.

Create a job by completing the following steps:

1. On the left navigational panel, click **Templates > Jobs**.
2. On the **Jobs** page, click **Create**.
3. In the **Create a Job** window, specify the name to be `Demo Job`.
4. Click **Save**.
   Result
   The **Main** tab for the new job is displayed.

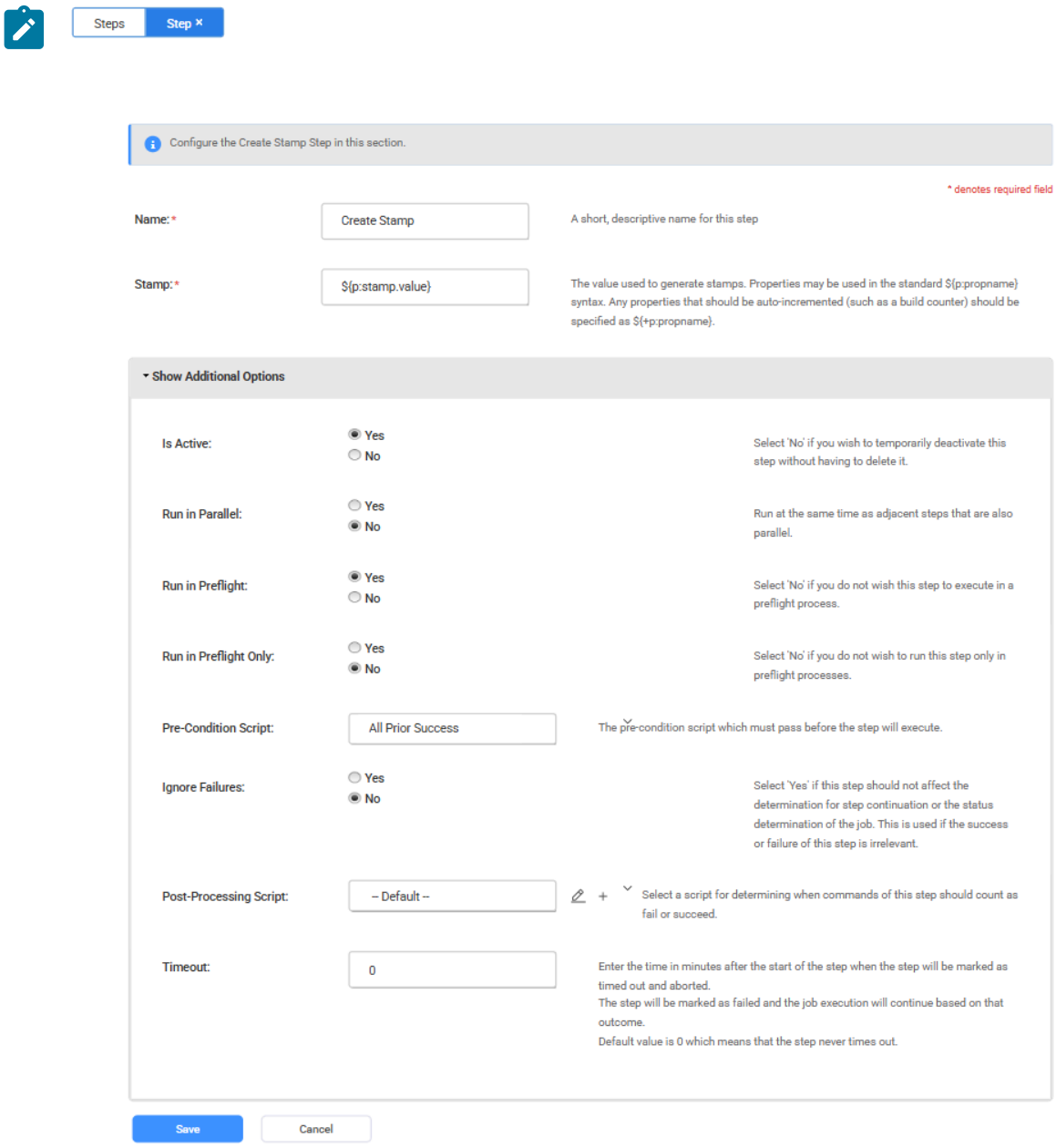
Results
Use the **Main** tab to add steps to the job. Except for the first step, steps can be inserted before or after other steps. When you run the job, the steps are run in the order you specify.

Creating a stamp step

A stamp is a custom identifier that is applied to a build. Stamps are often used in addition to the automatically generated build life identifiers. A stamp step is part of most jobs.

1. If the **Main** tab for the Demo Job is not displayed, then from the Build dashboard, click **Templates > Jobs**, and then click **Demo Job**.
2. Click **Create Step**.
3. Using the **Steps** dialog, click **Build**.
   The **Build** list expands to display the available build-type steps and plug-ins.
4. Click **Create Stamp** step, and then click **Select**.
   Result
   The Stamp dialog is displayed. The dialog that is displayed after selecting a step contains properties for the selected step, and other properties common to all steps.

   Note: Each step dialog has more options available than is initially displayed. To see the other options, click **Show Additional Options**. The additional options are the same for all steps and are set to default values that are acceptable for most jobs. The default value for the **Pre-Condition Script** field is `All Prior Success`, which means the step runs only if the preceding steps are successful. The additional options are show in the following figure. For most of the steps in the tutorial job, the default values are fine.



5. In the **Name** field, type `Create Stamp`.
6. In the **Stamp** field, enter a script to define the stamp: `1.0.0.${p:buildlife/id}`.
   The simplest way to make a unique stamp is to associate it with the build-life number. This short script looks-up the current build-life ID and applies it to the stamp. This script references a property that is part of every build life. Another option might be to reference a property that you can create and define later. For the tutorial, the build life property is fine. For information about the available properties, see the Scripting API Reference *server_url*`/tools/scripting/ibm-ucb-scripting/`.
7. Click **Save**.

Results

The Create Stamp step is listed on the **Main** tab for the job, as shown in the following figure:

After a job has a step, the **Create Step** button is no longer used and actions are used instead to insert steps.

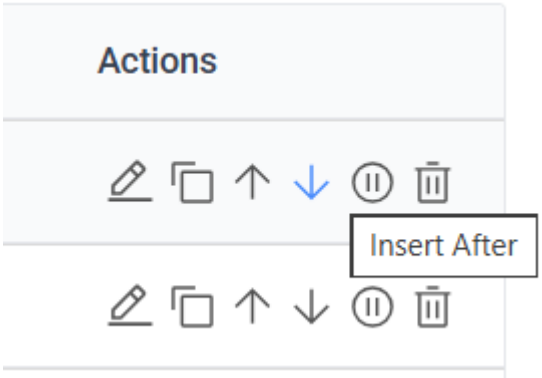
Cleaning the workspace

To ensure that no files from a previous build get incorporated into the new one, add a step to clean the agent's workspace. This step is part of most build jobs.

1. Click the **Insert After** action for the Create stamp step.

   Result

   After a job has its first step, steps are inserted before or after other steps. The **Insert After** action, as the name implies, inserts the new step after the selected step.

2. On the **Steps** dialog, click **Source** and then click **Clean Workspace**.
3. Click **Select**.

   Result

   The Clean Workspace step is selected.
4. In the **Name** field, type `Clean`.
5. Click **Save**.

Results

The Clean step is added to the job after the Create Stamp step, as shown in the following figure:

Steps

	Name	Type	Pre-Condition	Parallel	Actions
::	Creating Stamp	Create Stamp	All Prior Success	false	
::	Clean	Clean Workspace	All Prior Success	false	

Populating the workspace

This step places the checked-out code (defined later in the project's source configuration) in the agent's workspace.

1. Use the **Insert After** action for the Clean step.
2. On the **Steps** dialog, click **Source** and then click **Populate Workspace**.
3. Click **Select**.
4. In the **Name** field, type `Populate`.
5. Click **Save**.

Results

The Populate step is added to the job, as shown in the following figure

Steps

	Name	Type	Pre-Condition	Parallel	Actions
::	Creating Stamp	Create Stamp	All Prior Success	false	
::	Clean	Clean Workspace	All Prior Success	false	
::	Populate	Populate Workspace	All Prior Success	false	

Creating a get changelog step

This step retrieves source changes since the last build. The prior build life is located by using status or stamp parameters.

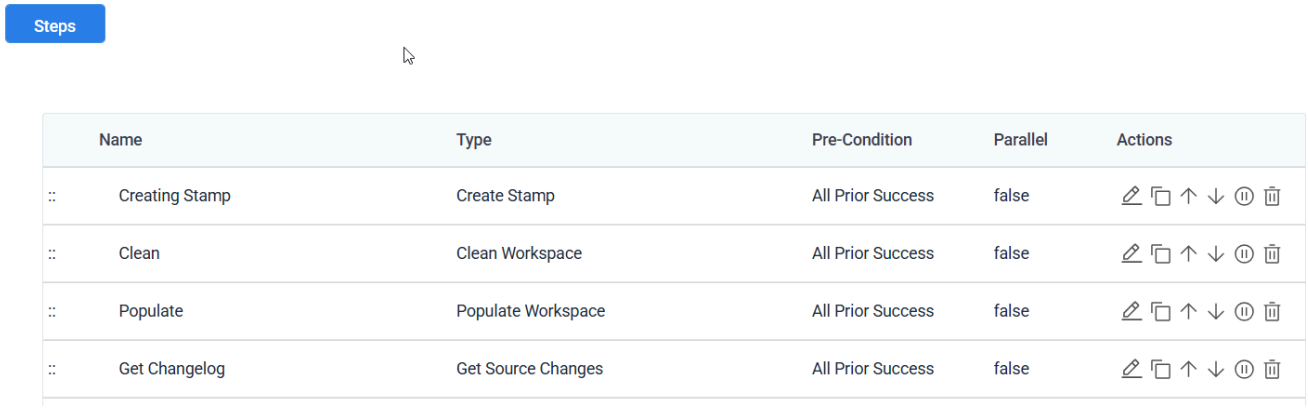
1. Click the **Insert After** action for the Populate step.
2. On the **Steps** dialog, click **Source** and then click **Get Source Changes**.
3. Click **Select**.
4. In the **Name** field, type `Get Changelog`.
5. Select **Success** on the **Start Status** list.
   The value that is selected determines which source items are retrieved. In this case, only items with a value of `Success` are retrieved.
6. Ensure that **Save Changes in Database** is selected (it is by default diabled).

This parameter can be used in build reports.

7. Click **Save**.

Results

The Get Changelog step is inserted into the job after the Populate step, as shown in the following figure:



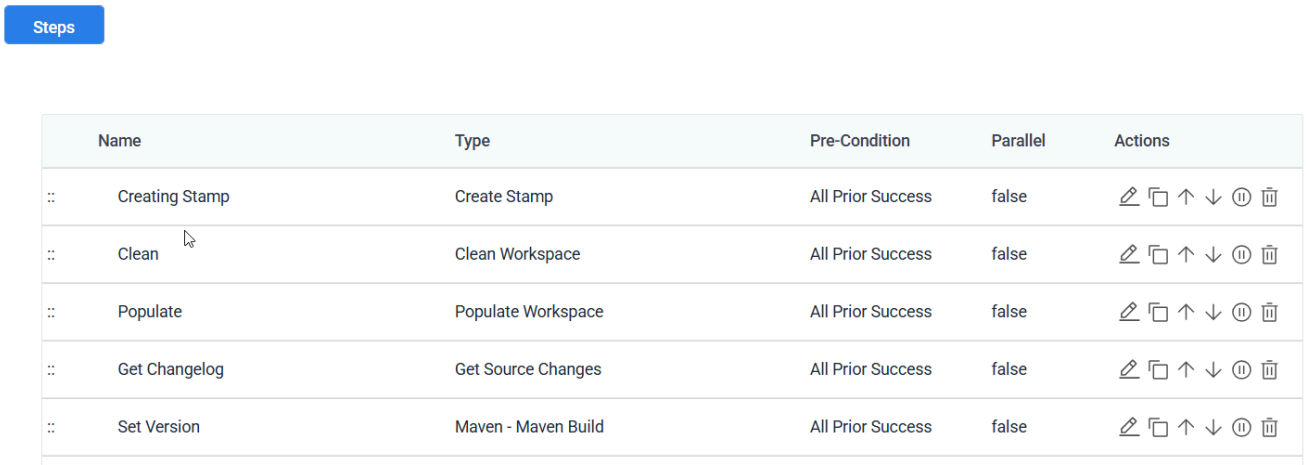
Creating the set version step

Create a version ID that Maven can use.

1. Click the **Insert After** action for the Get Changelog step.
2. On the **Steps** dialog, click **Build > Build Tools > Maven**, and then click **Maven Build**.
3. In the **Name** field, type `Set Version`.
4. Ensure that the default value `pom.xml` is set in the **Maven POM File** field.
   Maven uses the project object model file (POM) to control builds.
5. In the **Goals** field, type `versions:set`.
   Goals are similar to Ant tasks and define the actions that Maven performs.
6. **Optional:** If you did not put the Maven home variable on your system PATH variable, enter it in the **Maven Home** field.
7. In the **Properties** field, type `newVersion=${p:buildlife/latestStamp}`.
8. Click **Save**.

Results

The Set Version step is inserted into the job after the Get Changelog step, as shown in the following figure:



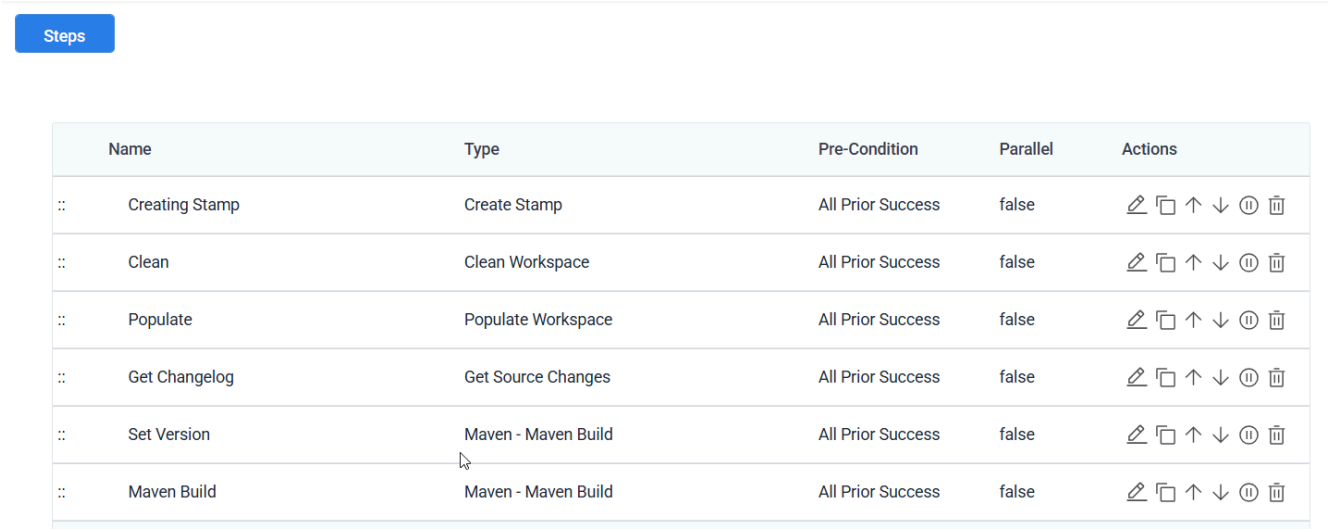
Creating the Maven build step

This step uses Maven to build the project.

1. Click the **Insert After** action for the Set Version step.
2. On the **Steps** dialog, click **Build > Build Tools > Maven**, and then click **Maven Build**.
   This step uses the same plug-in step as the previous step, Set Version.
3. In the **Name** field, type `Maven Build`.
4. Ensure that the default value `pom.xml` is set in the **Maven POM File** field.
5. In the **Goals** field, type `${p:maven.goals}`.
   Unlike the other properties you used in the job, this value references a new property. Later, you create the property and define the value.
6. **Optional:** If you did not put the Maven home variable on your system PATH variable, enter it in the **Maven Home** field.
7. Click **Save**.

Results

The Maven Build step is inserted into the job after the Set Version step, as shown in the following figure:



Creating the publish unit tests step

JUnit is a testing framework for Java™. This step runs tests and reports the results. It is configured to always run.

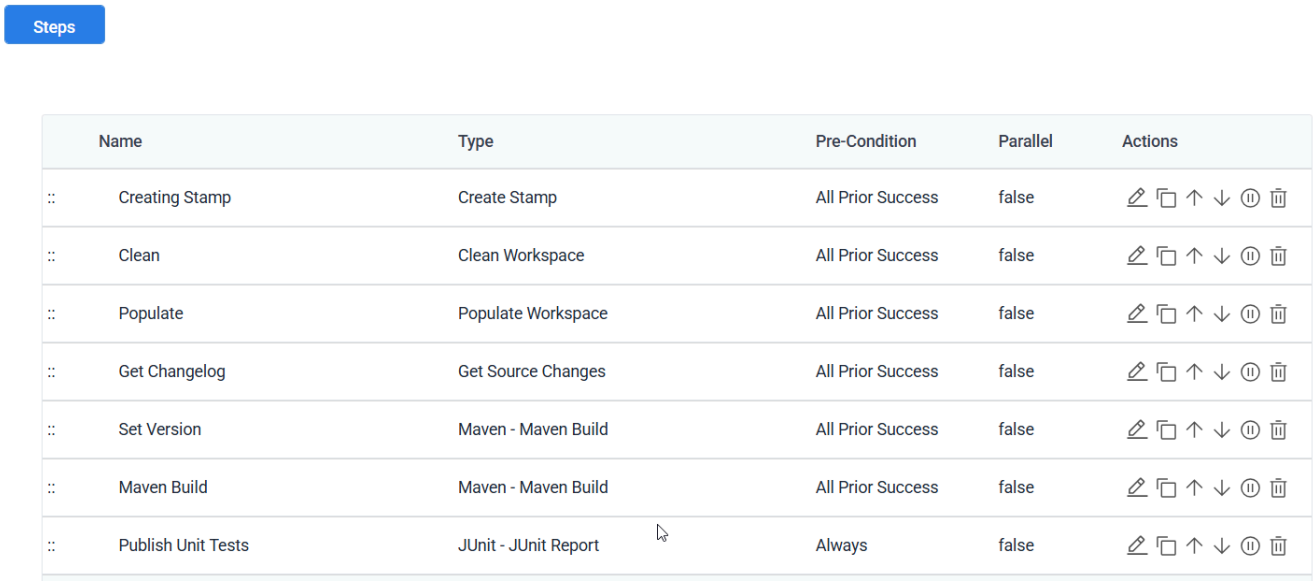
Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the tutorial system on page 246.

1. Click the **Insert After** action for the Maven Build step.
2. On the **Steps** dialog, click **Quality > Unit Testing > JUnit**, and then click **JUnit Report**.
3. In the **Name** field, type `Publish Unit Tests`.
4. In the **Report Name** field, type `JUnit Report`.
5. In the **Source Directory** field, type `target/surefire-reports`.
6. In the **Include Patterns** field, type `TEST*.xml`.
7. Display the **Pre-Condition Script** list by clicking **Show Additional Options**, and then select **Always**.
   Unlike the other steps in this job, this step always runs even if the preceding step fails. Test reports are displayed on the **Tests** tab for a selected build life. Reports are also written to the location you specified with the **Source Directory** field. Test files that match the pattern that is specified in the **Include Patterns** field are written to the same directory.
8. Click **Save**.

Results

The Publish Unit Tests step is inserted into the job after the Maven Build step, as shown in the following figure:



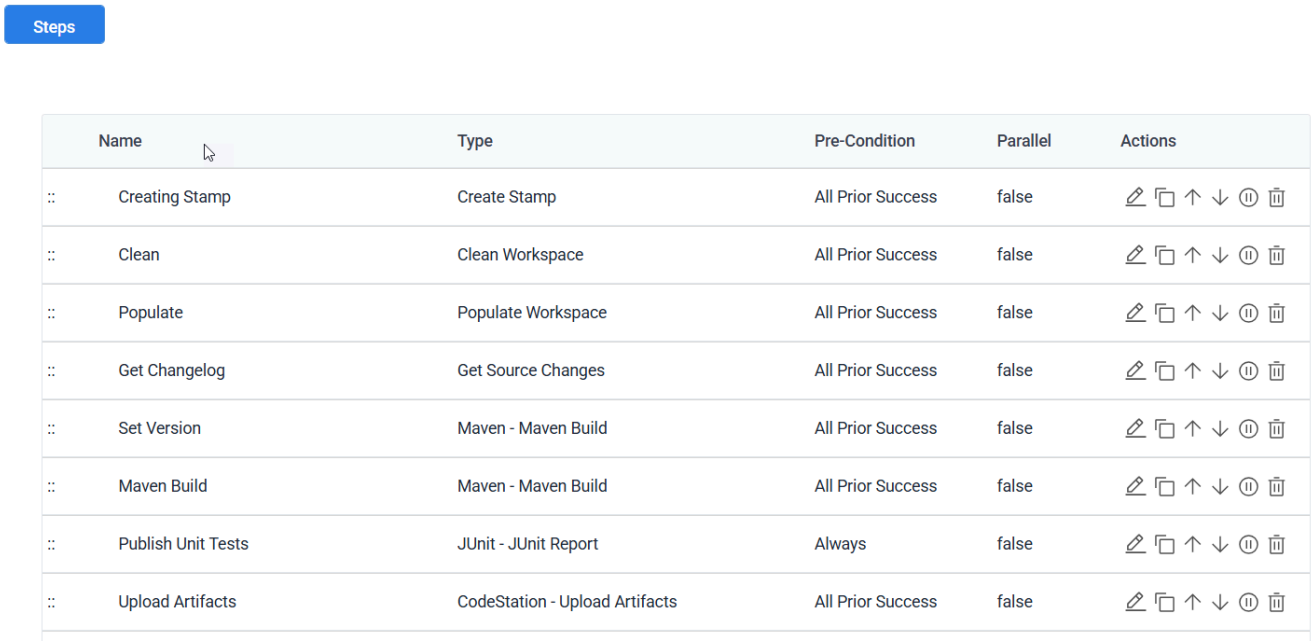
Creating the upload artifacts step

Upload the build artifacts to the CodeStation repository.

1. Click the **Insert After** action for the Publish Unit Tests step.
2. On the **Steps** dialog, click **Build Systems**, and then click **CodeStation**.
3. Click **Upload Artifatcs**, and then click **Select**.
4. In the **CodeStation-Upload Artifacts** window, in the **Name** field, type `Upload Artifacts`.
5. Click **Save**.

Results

The Upload Artifacts step is inserted into the job after the Publish Unit Tests step, as shown in the following figure:



Creating the assign success step

If all previous steps are successful, this step assigns a status of Success to the entire job. If the preceding steps are not successful, this step does not run and the next step runs instead.

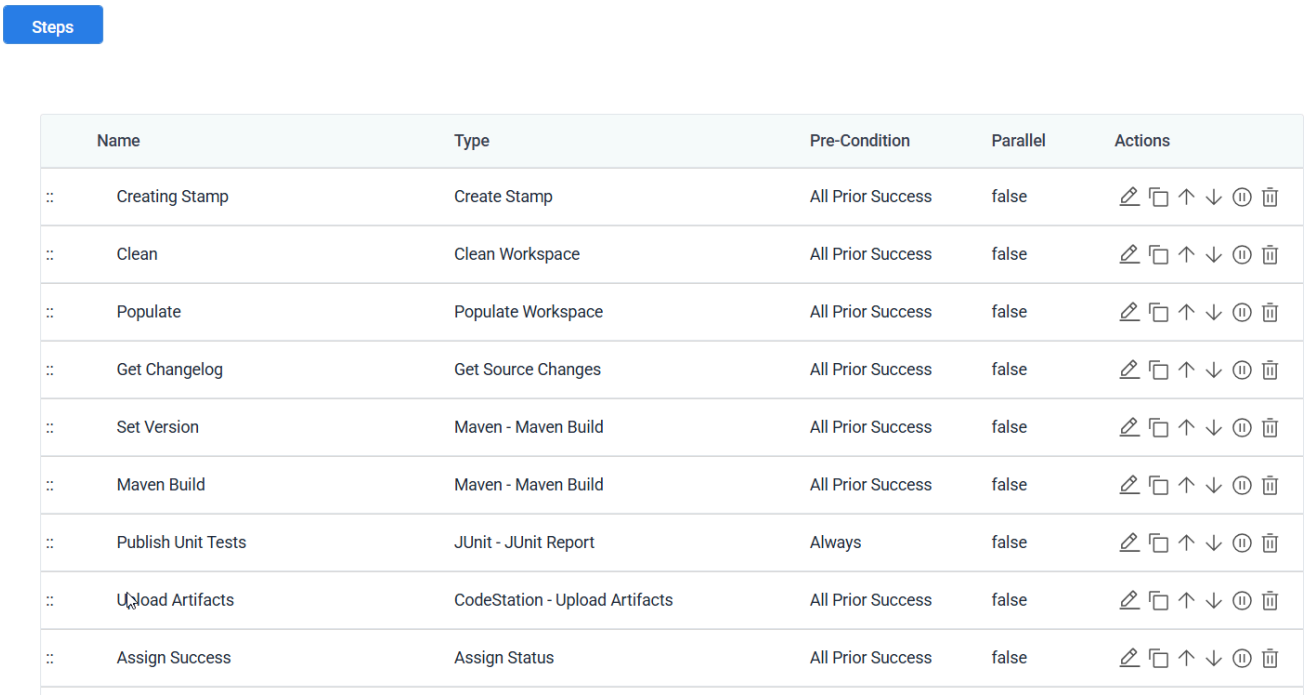
1. Click the **Insert After** action for the Upload Artifacts step.
2. On the **Steps** dialog, click **Build**, and then click **Assign Status**.
3. In the **Name** field, type `Assign Success`.
4. In the **Status** list, select **Success**.
5. Click **Save**.

   Result

   Because the **Pre-Condition Script** field is set to **All Prior Success** by default, you can accept the value.

Results

The Assign Success step is inserted into the job after the Publish Unit Tests step, as shown in the following figure:



Creating the assign failure step

This step assigns a status of Failure to the job. The step runs if any of the preceding steps fail.

1. Click the **Insert After** action for the Assign Success step.
2. On the **Steps** dialog, click **Build**, and then click **Assign Status**.
3. In the **Name** field, type `Assign Failure`.
4. In the **Status** list, select **Failure**.
5. Display the **Pre-Condition Script** list, and select **Any Prior Failure**.
6. Click **Save**.

Results

The Assign Failure step is inserted into the job after the Assign Success step.

Lesson checkpoint

In this lesson, you created job and added the steps required to build JUnit.

About this task

This job builds JUnit and assigns a status to the job. It also runs the JUnit plug-in step to create a report about the job. A job can use steps from any installed plug-in. Steps have preconditions that determine whether they run. Steps are configured individually as you insert them into a job.

The steps for the completed Demo Job are shown in the following figure:

Steps

	Name	Type	Pre-Condition	Parallel	Actions
::	Creating Stamp	Create Stamp	All Prior Success	false	
::	Clean	Clean Workspace	All Prior Success	false	
::	Populate	Populate Workspace	All Prior Success	false	
::	Get Changelog	Get Source Changes	All Prior Success	false	
::	Set Version	Maven - Maven Build	All Prior Success	false	
::	Maven Build	Maven - Maven Build	All Prior Success	false	
::	Publish Unit Tests	JUnit - JUnit Report	Always	false	
::	Upload Artifacts	CodeStation - Upload Artifacts	All Prior Success	false	
::	Assign Success	Assign Status	All Prior Success	false	
::	Assign Failure	Assign Status	All Prior Success	false	

Lesson 3: Creating a project template

Projects manage builds, dependencies, and artifacts. A project template is a reusable blueprint that defines a generic project.

About this task

Because project templates are reusable, you can apply the same standards to similar projects. Properties can be defined and the values supplied later by the projects created from the templates. A project template might always use a particular type of build script, for example, but the name or location of the script can vary by project.

In this lesson, you create a project template that you use later to create the tutorial project.

1. On the left navigational panel, click **Templates**.
   The **Projects** tab is open by default .
2. Click **Create**.
3. In the **Name** field, type `Demo Project Template`.
4. Click **Save**.
   Result
   The **Main** tab for the new project template is displayed.
5. Click **Build Pre-Processing**, and click **Edit.**.
6. Click **Agent Pool Selection**, and then select the agent pool you created when you configured the tutorial system on page 246.
7. Click **Save**.

Lesson checkpoint

In this lesson, you created the project template you use to build the tutorial project.

About this task

For more information about project templates, see Creating project templates on page 384.

Lesson 4: Creating a source template

A source template is a reusable source configuration that can be used by any project that requires the same source type.

Before you begin

Ensure that you installed the Git source-type plug-in before you start this lesson.

About this task

In this lesson, you define a Git configuration that is used by the tutorial project.

1. On the left navigational panel, click **Templates**, click **Source**, and then click **Create**.
2. In the **Source Type** list, select **Git**.
   All installed source-type plug-ins are listed.
3. Click **Select**.
   Result
   Fields that are related to the selected source type are displayed.
4. In the **Name** field, type `GitHub`.
5. In the **Working Directory Script** list, select **Default Workflow Directory**.
6. In the **Remote URL** field, type `${p:git.project}`.
   This refers to a property you create in the next section. You supply the property value when you configure the project.
7. In the **Branch** field, type `${p?:git.branch}`.
8. Click **Save**.

Results

The **GitHub** source template is listed along with the other source templates on the **Source Templates** tab.

Creating properties for the source template

Define the properties you referenced in the previous section.

1. If the **Source** tab is not already displayed, then on the Build dashboard, click **Templates**, click **Source**, and then click the `GitHub` source template that you created earlier in the lesson.
2. Click **Properties**, and then click **Create**.
3. Create the `git.project` property by completing these steps.
   a. In the **Name** field, type `git.project`.
   b. In the **Label** field, type `Git Project`.

The label text identifies the property to users that use the source template.

c. In the **Description** field, type `Path of the GitHub project within the account`.
The description text is displayed to users.

d. In the **Display Type** list, select **Text**.

e. In the **Value** area, accept the default option **Defined**.
The **Value** option determines when and where the value for the property is defined. By selecting **Defined**, you can provide a value in the **Default Value** field, or supply the value on the project configuration before the project runs.

f. Leave the **Default Value** field blank.
You define the property value when you configure the project.

g. Select the **Value Required** check box.

h. Click **Save**.

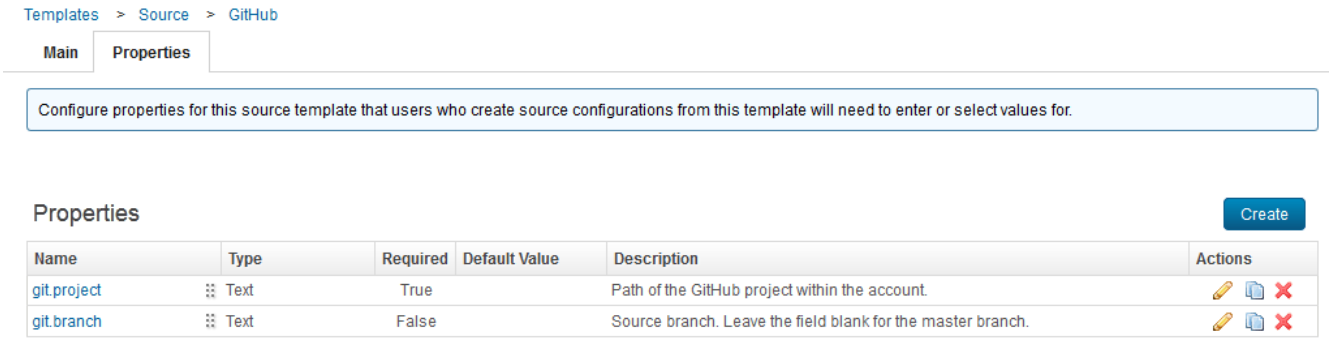
4. Create the `git.branch` property by completing these steps.
   a. On the **Properties** tab, click **Create**.
   b. In the **Name** field, type `git.branch`.
   c. In the **Label** field, type `Git Branch`.
   d. In the **Description** field, type `Source branch. Leave the field blank for the master branch`.
   e. In the **Display Type** list, select **Text**.
   f. Leave the **Default Value** field blank.
   g. Clear the **Required** check box.
   h. Click **Save**.

Lesson checkpoint

In this lesson, you created a source template and defined properties that identify the source material used in the build.

About this task

The GitHub source template properties are shown in the following figure:



Lesson 5: Creating a process template

Build a process template for the tutorial project.

About this task

In this lesson, you create a process template, define a process, create a property, and an artifact set that is used by the project.

1. On the left navigational panel, click **Templates**, click **Process**, and then click **Create**.
2. In the **Name** field, type `Demo Build Process`.
3. Under **Type**, make sure that **Build Process** is selected.
4. Click **Save**.

Results

The **Main** tab for the new process template is displayed.

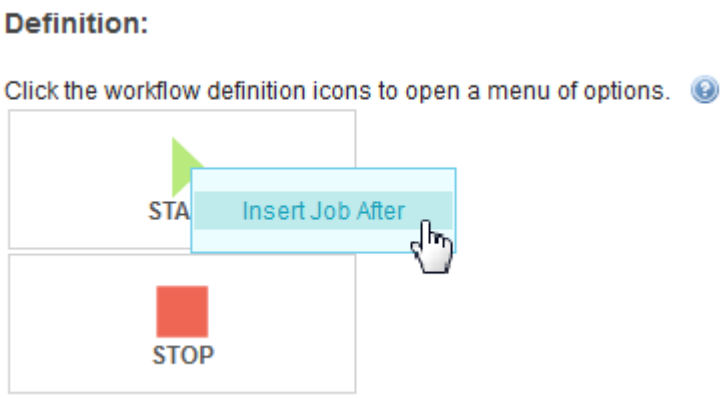
Creating a process definition

Process definitions identify the jobs that comprise a build.

About this task

In order for a job to be used in a project, it must be assigned to the process. A process can have several jobs and their order in the process is determined similarly to the way steps are inserted into jobs. In this lesson, you assign the job you created in the last lesson to a process definition.

1. On the Build dashboard, click **Templates > Process**.
2. In the **Process Templates** page, click **Demo Build Process**, and then click **Definition**.
3. Click **Start**, and then select **Insert Job After**.
   The **Insert Job After** action is displayed after you click **Start**, as show in the following figure:



4. In the **Insert Start Job** window, in the **Job** list, select the job you created in the last lesson, **Demo Job**.
5. In the **Pre-condition** list, select **Always**.

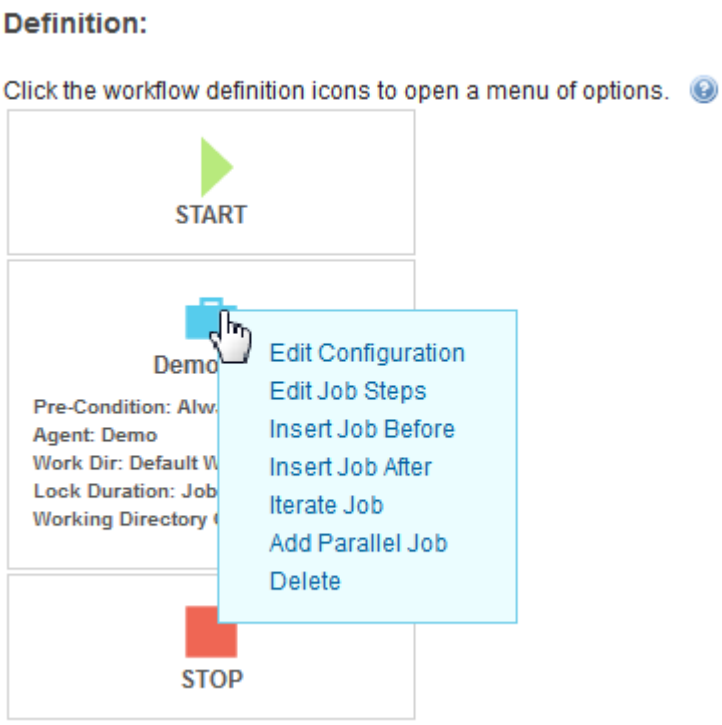
6. In the **Agent Selection** area, select **Agent Pool Selection**, and then select the agent pool you created when you configured your tutorial system on page 246.
   An agent in the agent pool runs the jobs assigned to the workflow.
7. In the **Working Directory** list, select **Default Workflow Directory**.
   An agent's default working directory is:

   ```
   agent_directory/var/work/project/
   ```

8. Click **Insert Job**.

Results

After a job is added to the process definition, the list of available process actions increases. Actions to insert jobs before or after other jobs, or to delete jobs are available, as show in the following figure.



Creating process template properties

Define the property you referenced in Lesson 2.

About this task

In this step, you define the property you referenced in the Maven Build step on page 254, `${p:maven.goals}`, in Lesson 2.

1. Using the Demo Build Process, click **Properties**, and then click **Create**.
2. In the **Name** field, type `maven.goals`.
3. In the **Label** field, type `Maven Goals`.
4. In the **Display Type** list, select **Text**.

5. In the **Default Value** field, type `package`.
   This is a Maven lifecycle build phase.
6. Select the **Value Required** check box.
7. Click **Save**.

Creating process artifact sets

Generated artifacts are assigned to an artifact set. An artifact set is a collection of build artifacts.

Before you begin

Ensure that you configured a Maven artifact set on page 246 before you start this step.

About this task

In this lesson, you configure the artifact set that is used by the project.

1. Ensure that the window for your build process is displayed and click **Artifacts**, and then click **New Artifact Config**.
2. In the **Artifact Set** list, select the artifact set you created earlier, **maven**.
3. In the **Base Directory** field, type `target`.
4. In the **Include Artifacts** field, type `*.jar`.
5. Click **Save**.

Lesson checkpoint

In this lesson, you created a process template and added a property and an artifact set to it.

Lesson 6: Creating a project

You use projects to configure and run builds. After a build runs, projects provide a wealth of information that you can use to manage the project lifecycle and troubleshoot problems.

Before you begin

Ensure that you completed all of the preceding steps before you start this lesson.

About this task

Projects are built from templates. In this lesson, you will use the templates that you made earlier to create the tutorial project. You can run builds automatically on a predetermined schedule, or you can run builds manually, as you do in the tutorial. To define the tutorial project, you select the process that you defined earlier. When you start the build, the workflow begins running the job that you created with the job template. The project determines the source code to use, and manages any dependencies. Each build generates a unique build life ID.

1. On the left navigational panel, click **Projects**, and then click **Create**.
2. Select the project template that you created in Lesson 3 on page 259, **Demo Project Template**.
3. In the **Name** field, type `Demo Project`.
4. Click **Save**.

Configuring the project build process

A project must be associated with a build process. A project might have any number of build processes associated with it.

1. Ensure that the **Configuration** tab for the Demo Project project is displayed, then, under **Build Processes**, click **New**.
2. In the **Template** list, select **Demo Build Process**, and then click **Select**.
   The **Maven Goals** field is derived from the process template that you created in last lesson.
3. In the **Name** field, type `Demo Build`.
4. In the **Maven Goals** field, accept the value, `package`.
   The `package` value is a basic Maven lifecycle build phase. The `package` attribute instructs Maven to put the compiled code into a distributable format, such as a JAR file. You can use the tutorial project to run other Maven builds and change the goals.
5. Click **Save**.

Results

The **Configuration** tab for the new build process is displayed.

Configuring the project source

A project's build source is defined by assigning a source template to a build configuration.

Before you begin

Ensure that the **Configuration** tab for the build process you defined in the previous section is displayed.

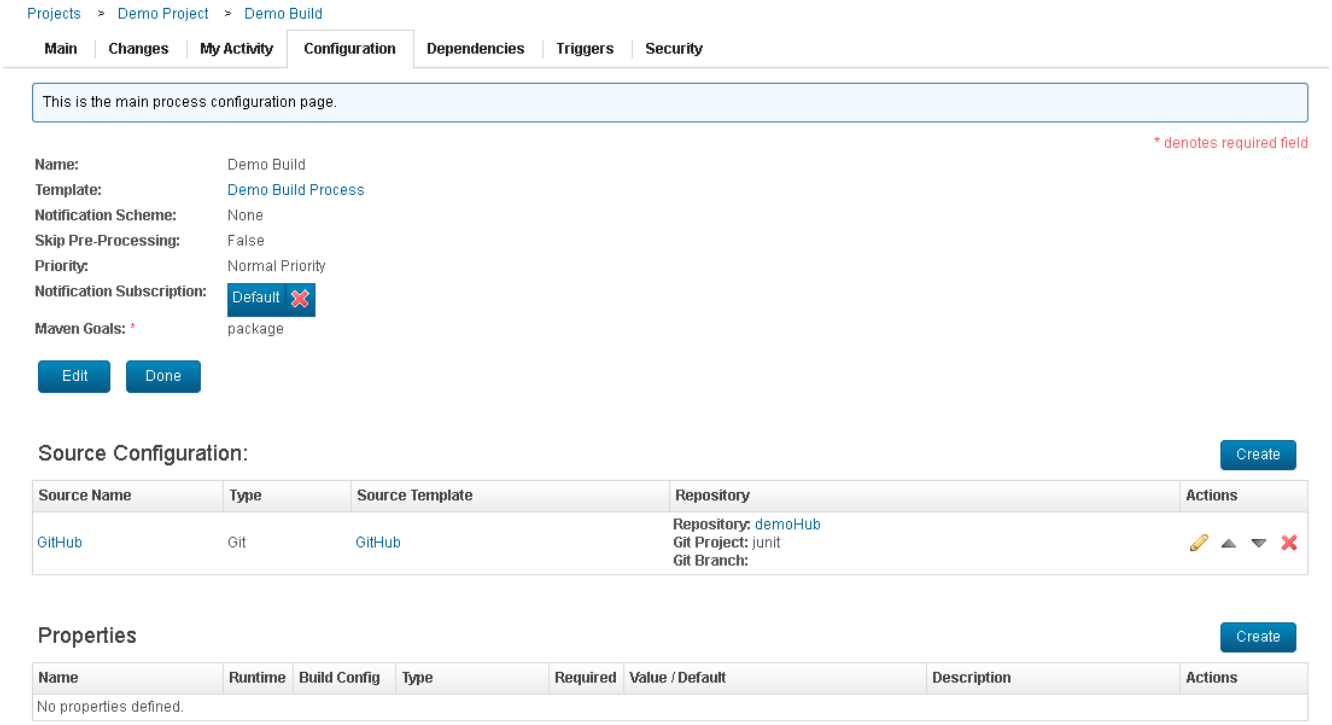
About this task

You define the source configuration by selecting the source template that you created earlier, and by using template properties to identify the source location.

1. Under **Source Configuration**, click **Create**.
2. In the **Source Template** list, select the source template that you created earlier, **GitHub**, and then click **Select**.
3. In the **Name** field, type `GitHub`.
4. In the **Repository** list, select the repository that you created earlier, **demoHub**.
5. In the **Git Project Path** field, type `junit`.
   Because the project uses the `master` branch, you can leave the **Git Branch** field blank.
6. Click **Save**.

Results

With the build process defined and the build source identified, the project is complete. The finished Demo Build process is shown in the following figure:



Lesson checkpoint

In this lesson, you created a project and defined a build process and build source.

About this task

The project is constructed from the templates you made earlier in the tutorial. All projects are built from templates; when you create a project, you identify the templates and provide property values for them. A project can inherit properties from any of the templates used in the project.

Lesson 7: Running the build

Run the project build process to transform the source code into the finished application, JUnit.

Before you begin

Ensure that you completed all previous steps in the tutorial.

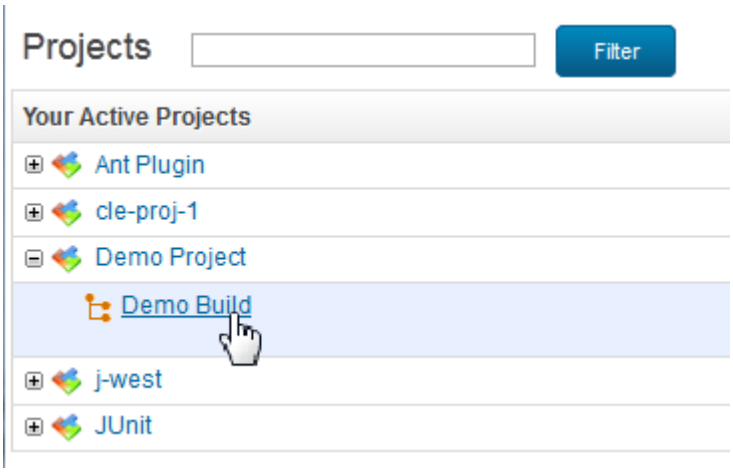
About this task

You run a build by running a build process that is defined for the project. The build process takes the source material and transforms it, using a job that is assigned to the project, into the final output.

1. On the left navigational panel, click **Projects**, and then expand the project that you created in the last lesson, **Demo Project**.

   Result

   Builds configured for the selected project are listed beneath the project, as shown in the following figure:



2. Click **DemoBuild**.

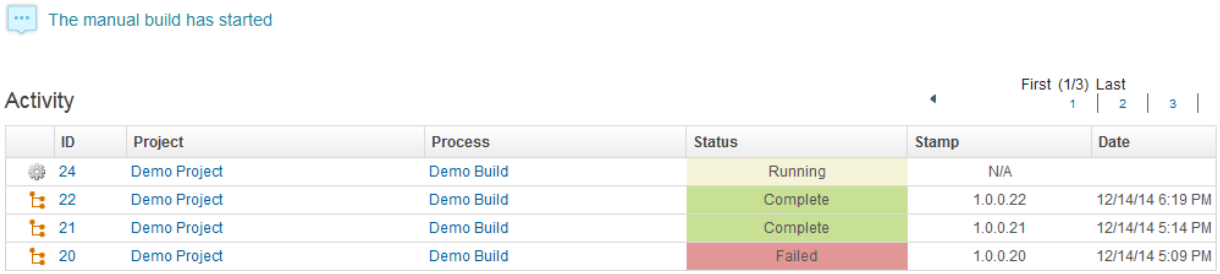
   Result

   The **Main** tab for the DemoBuild project build is displayed.

3. Click **Build**.

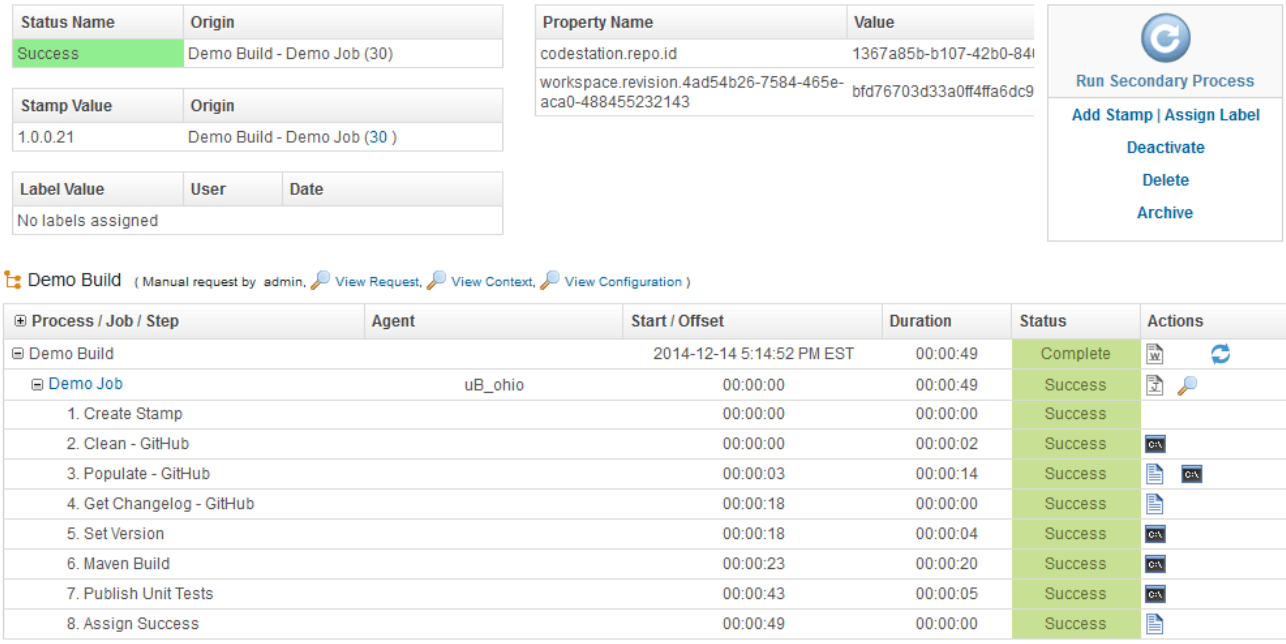
   Result

   The build process starts and the **My Activity** tab for the build is displayed. The build, with a new build life number, is listed with a status of `Running`, as shown in the following figure:



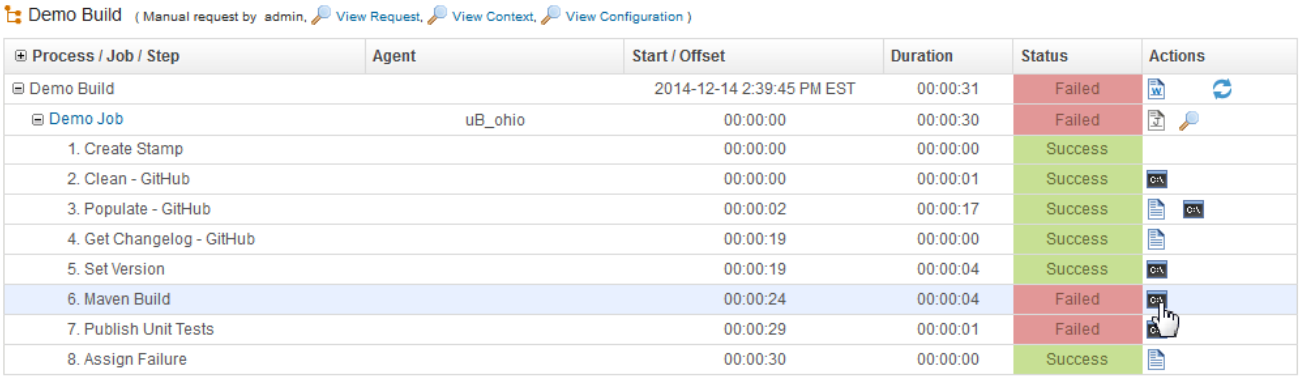
Results

Refresh the **My Activity** page. If the build is successful, the status changes to `Complete`; otherwise, the status changes to `Failed`. A successful run is shown in the following figure:



What to do next

The project's Main tab contains information about the builds generated by the project. To see details about a build, click the build life ID in the **Build** column. To see a step's output, use the **View Output** action, as shown in the following figure. Use the log to identify errors and troubleshoot failed steps.



To see the JUnit test results, display the **Test** tab for the build.

Lesson checkpoint

In this lesson, you ran a build of the tutorial project, Demo Project, and examined its progress, tests, and final output.

About this task

In this lesson, you ran a build for the project you created. The build created the application JUnit. The JUnit application is in the *agent_install*`\var\work\project\Demo_Project\Demo_Build` folder. Project artifacts are stored in the CodeStation repository.

Summary

In this tutorial, you learned to build a project and run a build with Build.

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How to create a repository and use it in a build process
- How to create jobs and add steps to them
- How to create source and process templates
- How to create project templates
- How to create projects and configure their sources, and processes
- How to run a build and examine its tests and output

Create a project that integrates with DevOps Deploy

In this tutorial, you create a project that builds distributable artifacts and loads them into HCL DevOps Deploy (Deploy) components. The source code is downloaded from GitHub and built with Ant.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a Git repository that connects to GitHub.
- Create templates that define properties. You specify values for these properties when you create the project.
- Create a process definition and associate an agent pool with it.
- Create a job that uses Ant to build the project artifacts, run JUnit tests, create an Deploy component version, and assign a status to the component.
- Create a project that is based on the templates that you create.
- Run a build and monitor its progress.
- Examine the results of the build by reviewing the component version that was created by the project in Deploy.

Time required

2 hours

Introduction

In this tutorial, you create a project that builds distributable artifacts and loads them into HCL DevOps Deploy (Deploy) components. The source code is downloaded from GitHub and built with Ant.

The project that you create in this tutorial introduces you to the most-used product features. The workflow that you create contains the job types that you use every day.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a Git repository that connects to GitHub.
- Create templates that define properties. You specify values for these properties when you create the project.
- Create a process definition and associate an agent pool with it.
- Create a job that uses Ant to build the project artifacts, run JUnit tests, create an Deploy component version, and assign a status to the component.
- Create a project that is based on the templates that you create.
- Run a build and monitor its progress.
- Examine the results of the build by reviewing the component version that was created by the project in Deploy.

Time required

This tutorial takes approximately 2 hours to finish. If you explore other concepts that are related to this tutorial, it might take longer to complete.

System requirements

To run this tutorial, you need the following resources:

- An installation of Build, version 6.1 or later.
- An Build agent that is installed and configured. The agent must be able to access the Deploy server you use in the tutorial.
- An installation of Deploy.

Prerequisites

To complete this tutorial, you must have an account on the Build server with permissions to create and edit projects, process definitions, jobs, and repositories, and to create and run processes. In addition, you must have an account an the Deploy server with, at a minimum, permission to view, create, and edit component versions, and access the **Settings** page.

Configure the tutorial system

Before you can run the tutorial project, you must configure your Build and DevOps Deploy environments.

Before you begin

Before you begin the tutorial, make sure that you meet the Prerequisites on page 270.

About this task

To complete the tutorial, you must have an installation of Build and HCL DevOps Deploy (Deploy). Your Build agent must have access to the Deploy server.

To prepare the tutorial system.

1. Prepare a computer for use as the target system.
2. Install the plug-ins that are used in the tutorial.
   The tutorial uses several plug-ins. Most of them are installed when you install the product, but you must install some plug-ins yourself. Download and install these plug-ins:

- Git
- JUnit

You can download plug-ins from the Build plug-in site: Build plug-ins website. For information about installing plug-ins, see Installing plug-ins on page 351.

3. Typically, agents are not installed on the same computer where the Build server is located. For the tutorial, installing the agent on the computer where the server is located is fine.
4. Configure the agent on page 342.
   Your agent must be included in the list of configured agents on the **Agents** page with a status of `Online`.
5. Create an agent pool and add the installed agent to it. on page 345
6. Install the open source tool Git on the computer where the Build agent is located.
   Add the home directory for the tool to the *PATH* variable on the target system.

   Note: The tutorial also requires Ant. Ant is installed with the agent when the agent is installed.

7. Create the artifact set used by the tutorial by completing these steps:
   a. From the Build dashboard, click **System**, and then click **Artifact Sets**.
   b. In the **Artifact Sets** tab, click **Create New Artifact Set**.
   c. In the **New Artifact Set** dialog box, in the **Name** field, type `dist`.
   d. Click **Save**.
8. Install Deploy.
   You must know the version number of the server because the Deploy plug-in works differently for versions before 6.1. The Deploy server must be accessible by the Build agent. When the tutorial project runs, the agent communicates with the Deploy REST API.

   Note: The Deploy server must be running when you run the tutorial project.

9. Configure an integration on page 307 with the Deploy server you installed in the previous step. To complete the configuration, you need the URL of the Deploy server.
10. In your Deploy installation, create a component that is named Ant.
    When you create the component, provide the following property values:
    - **Component Type**: **Standard**
    - **Source Configuration Type**: **none**

    You can accept the default values for the other properties.

    Note: To complete this step, you need an Deploy account with, at a minimum, permission to view, edit, and create components. If you have your own component, you can use that. However, the tutorial uses the name of the component in one of its properties and you will have to substitute the name of your component for the name used in the tutorial. See Prerequisites on page 270.

11. In your Deploy installation, create a version status and name it Success.

The tutorial project assigns this status to the component version that it creates in Deploy. Make sure that you create a version status and not an inventory or snapshot status.

Note: To complete this step, you need an account with access to the Deploy **Settings** page. See Prerequisites on page 270.

Results

Your Build environment is ready to being the tutorial.

The tutorial project is also available in a file that you can download and import. To import the tutorial project, follow these steps:

1. Navigate your browser to the file repository URL:http://download.boulder.ibm.com/ibmdl/pub/software/rationalsdp/documentation/product_doc/UrbanCode/uBuild/.
2. Download the file named `Ant.xml`.
3. On the Build **Project** page, click **Import**, and then select the downloaded file.

   Note: Make sure that you install the plug-ins described earlier in this topic before you import the file.

4. In the **Import** window, click **Import**. The server imports the project. The server does not import plug-ins when it imports a project, so you still need to install the plug-ins described earlier in this topic.

Lesson checkpoint

The servers and agent are configured for the tutorial. If you installed the agent on a remote system, you must know the credentials that are required to access the remote system.

Lesson 1: Creating a Git repository

Repository integrations enable Build to check out code, access change logs, and do other project-related activities.

Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the tutorial system on page 270.

1. On the left navigational panel, click **System > Repositories**.
2. On the **Repositories** page, click **Create New Repository**.
3. In the **Source Plug-in** list, select **Git** and then click **Set**.
4. In the **Name** field, type `GitHub`.
5. In the **Repository Base URL** field, type `https://github.com/IBM-UrbanCode/ant`.
6. Click **Save**.

Lesson checkpoint

In this lesson, you created a repository that is based on the Git plug-in.

About this task

The GitHub repository is listed on the **Repositories** page.

For more information about repositories, see Repositories on page 330.

Lesson 2: Creating a job and adding steps

A job is a series of steps that, together, complete an activity, such as building a project. When a project runs, the steps run one at a time in the order that you specify.

Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the tutorial system on page 270.

About this task

In this lesson, you create a job and add steps to it that download source code, builds artifacts, runs tests, creates a component versions in HCL DevOps Deploy (Deploy), and assigns a status to the component. The job will be used by the workflow that you create later in the tutorial. As you create the job, you define several properties that will be used later when you create and run the project.

Creating a job

Most job steps are created from plug-ins. Plug-in steps can be thought of as distinct pieces of automation.

About this task

By combining steps from various plug-ins, you can create fully automated build processes. The number of steps that are provided by a particular plug-in can vary, and each step has a variable number of properties. Property values can be supplied when you define a step or at process run time.

Note: Several plug-ins are included in the product, and you can download other plug-ins as described earlier in the tutorial on page 270. Some steps, such as Assign Status and Create Stamp, are not derived from plug-ins, but you use them in the same way that you use plug-in steps.

Create a job by completing these steps:

1. On the left navigational panel, click **Templates > Jobs**.
2. On the **Jobs** page, click **Create**.
3. In the **Create a Job** window, name the job `Ant Build Job`.
4. Click **Save**.

   Result

   The **Main** tab for the new job is displayed.

Results

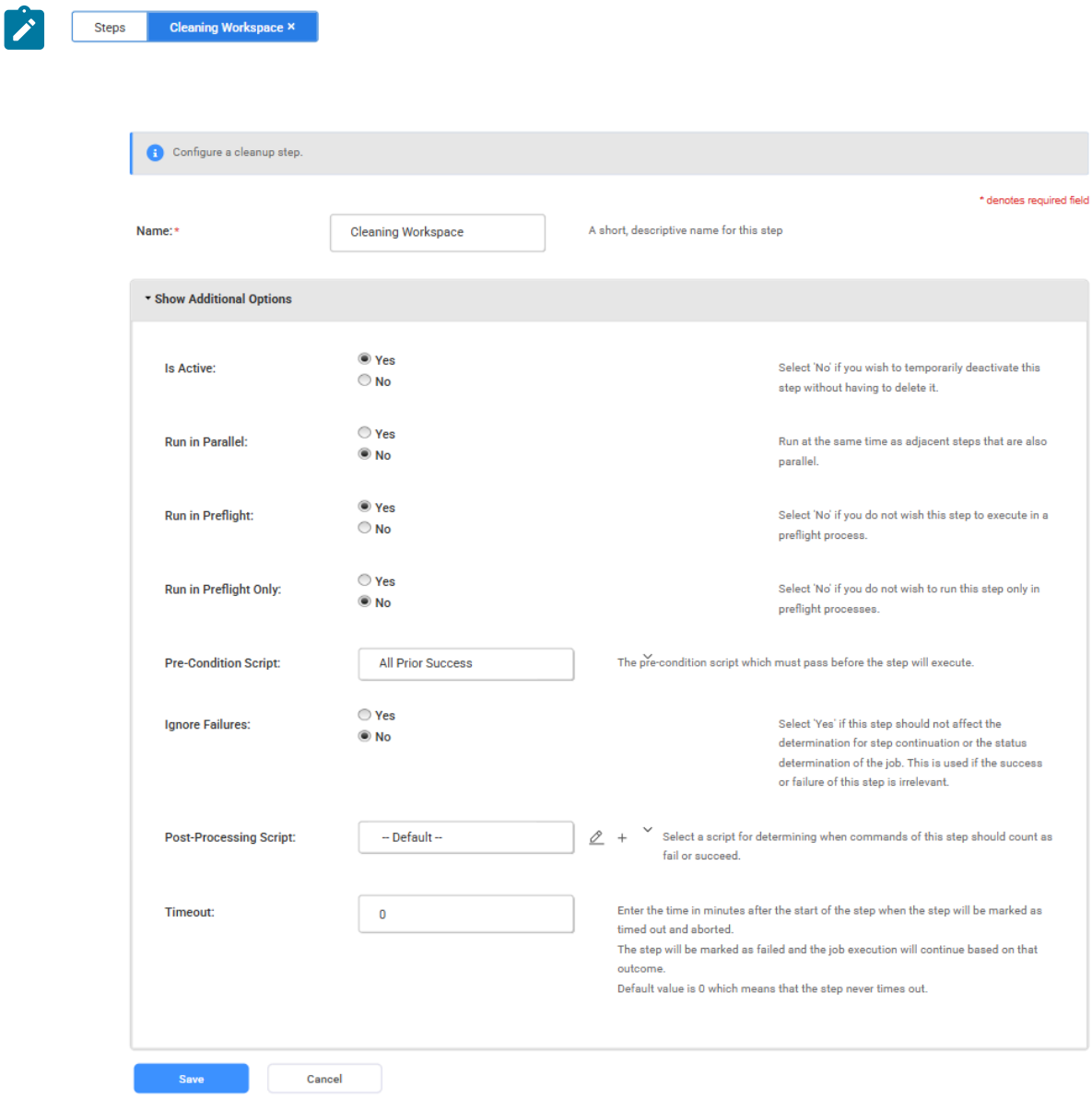
You use the job's **Main** tab to add steps to the job. Except for the first step, steps can be inserted before or after other steps. You can rearrange the steps and change their order. When you run the job, the steps are run in the order that you specify.

Cleaning the workspace

To ensure that no files from a previous build get incorporated into the new one, add a step to clean the agent's workspace. This step is part of most build jobs.

1. If the **Main** tab for the Ant Build Job is not displayed, then from the Build dashboard, click **Templates > Jobs**, and then click **Ant Build Job**.
2. Click **Create Step**.
3. Using the **Steps** dialog box, click **Source**.
   Result
   The **Source** list expands to display the available source-related steps and plug-ins.
4. In the **Steps** dialog box, click **Clean Workspace**, and then click **Select**.
   Result
   The **Change Source** dialog box is displayed. The dialog box that is displayed after you select a step contains options that are related to the selected step. Some of the options are unique to the step, while others are common to all steps.

   Note: Each step dialog box has more options available than are initially displayed. To see the other options, click **Show Additional Options**. The additional options are the same for all steps and are set to default values that are acceptable for most jobs. For example, the default value for the **Pre-Condition Script** field is `All Prior Success`, which means that the step runs only if the preceding steps succeed. The additional options are shown in the following figure. For most of the steps in this tutorial job, the default values are fine.



5. On the **Cleanup Source** dialog box, in the **Name** field, type `Cleanup`.
6. Click **Save**.

Results

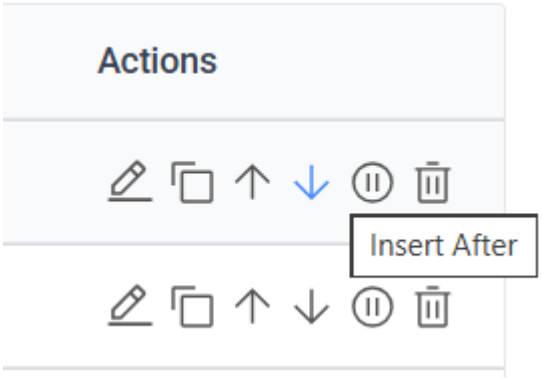
The Cleanup step is listed on the **Main** tab for the job.

After a job has a step, the **Create Step** push button is no longer used and actions are used instead to insert steps.

Populating the workspace

This step places the checked-out code, which is defined later in the project's source configuration, in the agent's workspace.

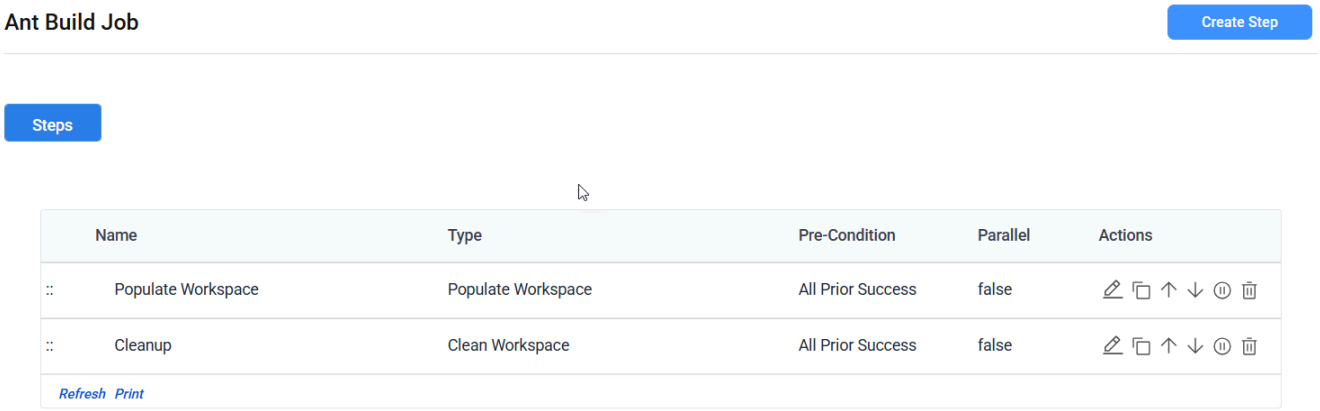
1. Use the **Insert After** action for the Cleanup step.



2. In the **Steps** dialog box, click **Source**, and then click **Populate Workspace**.
3. Click **Select**.
4. Using the **Populate Workspace** dialog box, in the **Name** field, type `Populate Workspace`.
5. In the **Date String** field, type the `${bsh:WorkspaceDate.get()}` BeanShell script line.
   This script sets the date to the date that the source is checked out. The default date is the `workspace.date` property on the job.
6. Click **Save**.

Results

The Populate Workspace step is added to the job, as shown in the following figure



Creating a stamp step

A stamp is an identifier that is applied to a build. Stamps are often used in addition to the automatically generated build life identifiers. A stamp step is part of most jobs.

1. If the **Main** tab for the AntBuild Job is not displayed, then from the Build dashboard, click **Templates > Jobs**, and then click **Ant Build Job**.
2. Use the **Insert After** action for the Populate Workspace step.
3. Using the **Steps** dialog box, click **Build**.

Result

The **Build** list expands to display the available build-type steps and plug-ins.

4. In the **Build** list, click **Create Stamp**, and then click **Select**.

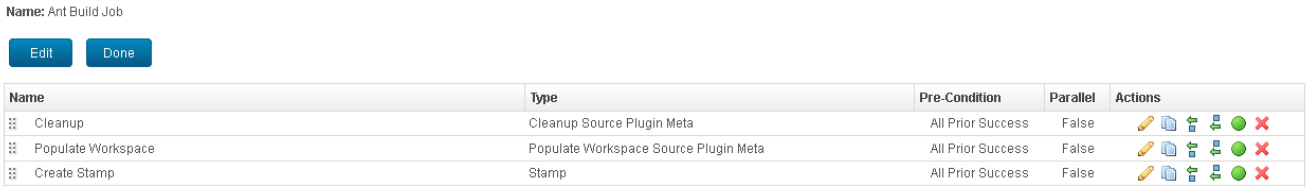
   Result

   The Stamp dialog box is displayed.
5. In the **Name** field, type `Create Stamp`.
6. In the **Stamp** field, enter a script to define the stamp: `${p:product.version}.${gvy:BuildLifeLookup.current.id}`.

   The Groovy script looks up the current build-life ID, which is part of every build life, and appends it to the `product.version` property. You create the `product.version` property later in the tutorial when you create the build process.
7. Click **Save**.

Results

The Create Stamp step is listed on the **Main** tab for the job, as shown in the following figure:



Creating a get changelog step

This step retrieves source changes that were saved since the last build. The prior build life is located by using status or stamp parameters.

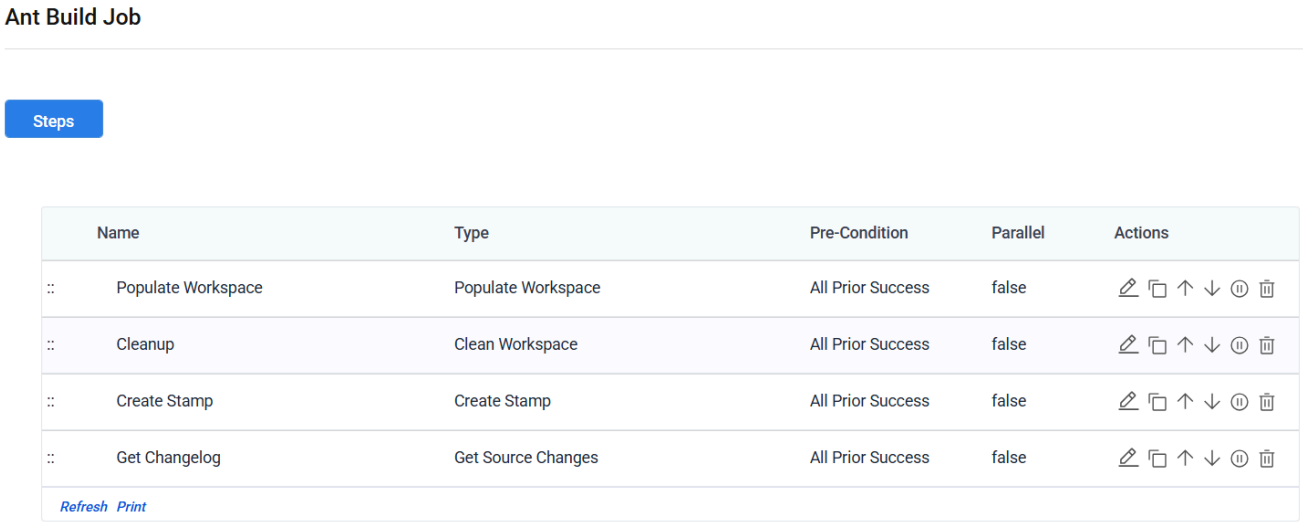
1. Click the **Insert After** action for the Create Stamp step.
2. In the **Steps** dialog box, click **Source**, and then click **Get Source Changes**.
3. Click **Select**.
4. In the **Get Changelog Source Plugin Meta** dialog box, in the **Name** field, type `Get Changelog`.
5. Select **Success** on the **Start Status** list.

   The value that is selected determines which source items are retrieved. In this case, only items with a `Success` value are retrieved.
6. Ensure that **Save Changes in Database** is selected, it is selected by default.

   This parameter makes the data available to the web application's user interface, and for reporting.
7. Click **Save**.

Results

The Get Changelog step is inserted into the job after the Populate step, as shown in this figure:



Creating an Ant build step

Create a step that runs an Ant build process.

About this task

This step runs an Ant build. Like most steps, the Ant build step is defined by the properties that you specify. Many properties that you define for this step are placeholders and their values will be specified by other templates or at run time.

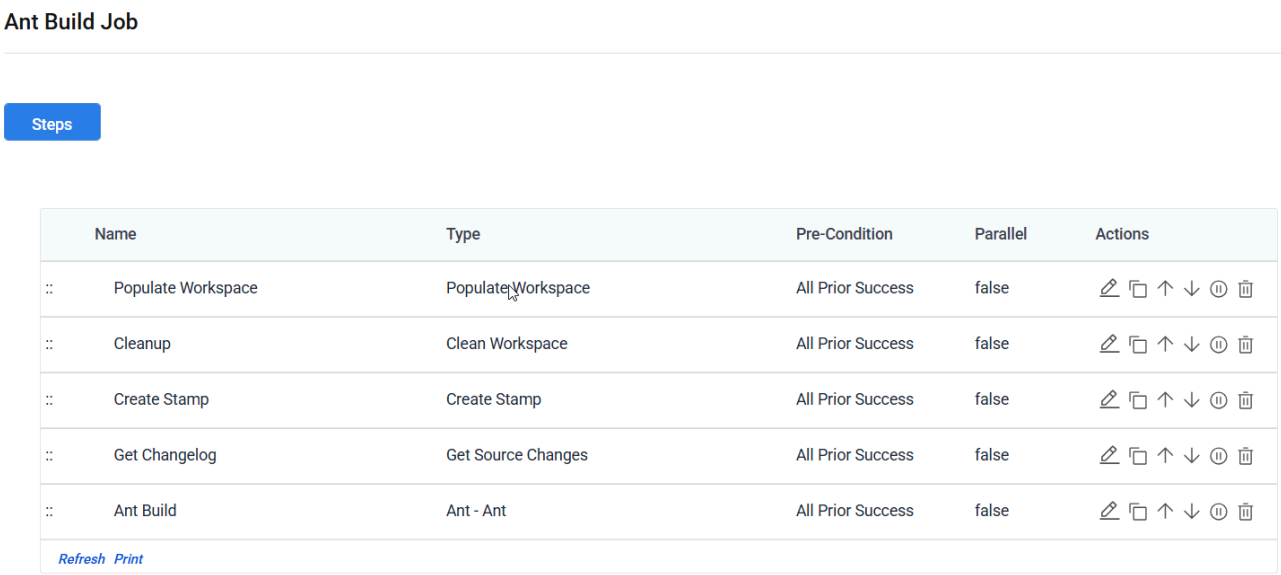
1. Click the **Insert After** action for the Get Changelog step.
2. In the **Steps** dialog box, click **Build > Build Tools > Ant > Ant**, and then click **Select**.
3. In the **Plugin Command** window, in the **Name** field, type `Ant Build`.
4. In the **Working Directory Offset** field, type `${p?:dir.offset}`.
   The question mark character, ?, can be used to reference properties. The format is `${p?:propertyName}`. In this instance, if the user specifies a value for the `dir.offset` property, this script returns the value. Otherwise, it returns an empty string.
5. In the **Ant Script File** field, type `${p:build.script.name}`.
6. In the **Targets** field, type `${p?:ant.targets}`.
7. In the **Properties** box, type `${p?:build.properties}`.
8. In the **Ant Properties** box, type `${p?:ant.properties}`.

   Note: The **ANT_HOME** and **JAVA_HOME** fields are set to default values for the computer where the agent is located.

9. Click **Save**.

Results

The Ant Build step is inserted into the job after the Get Changelog step, as shown in this figure:



Creating the publish JUnit tests step

JUnit is a testing framework for Java™. This step runs tests and reports the results.

Before you begin

Before you begin this lesson, make sure that you install the JUnit plug-in as described in Configure the tutorial system on page 270.

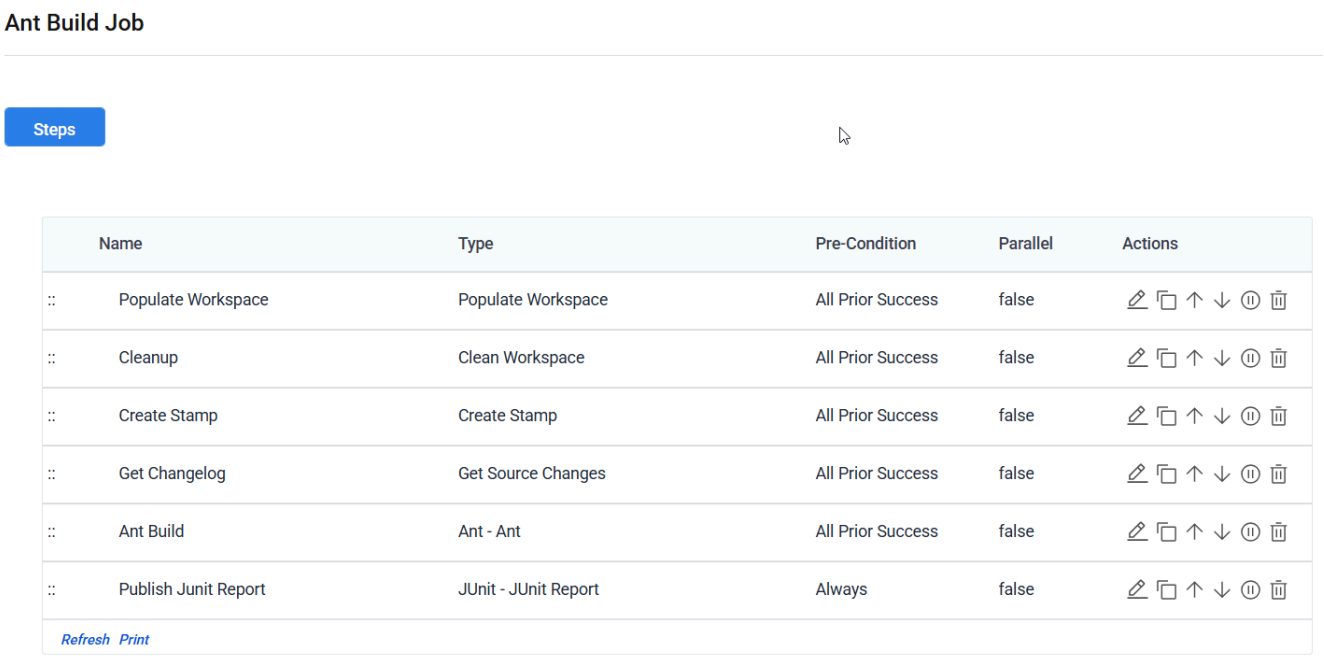
About this task

Unlike most of the other steps in the Ant Build job, this step is configured to always run even if the preceding step fails. Test reports are displayed on the **Tests** tab for build life.

1. Click the **Insert After** action for the Ant Build step.
2. In the **Steps** dialog box, click **Quality > Unit Testing > JUnit**, and then click **JUnit Report**.
3. Click **Select**.
4. In the **Name** field, type `Publish JUnit Report`.
5. In the **Report Name** field, type `JUnit Report`.
6. In the **Source Directory** field, type `${p:junit.report.base.dir}`.
   Reports are written to the specified location.
7. In the **Include Patterns** field, type `TEST*.xml`.
   Test files that match the specified pattern are written to the same directory.
8. Display the **Pre-Condition Script** list by clicking **Show Additional Options**, and then select **Always**.
9. Click **Save**.

Results

The Publish JUnit Report step is inserted into the job after the Ant Build step, as shown in this figure:



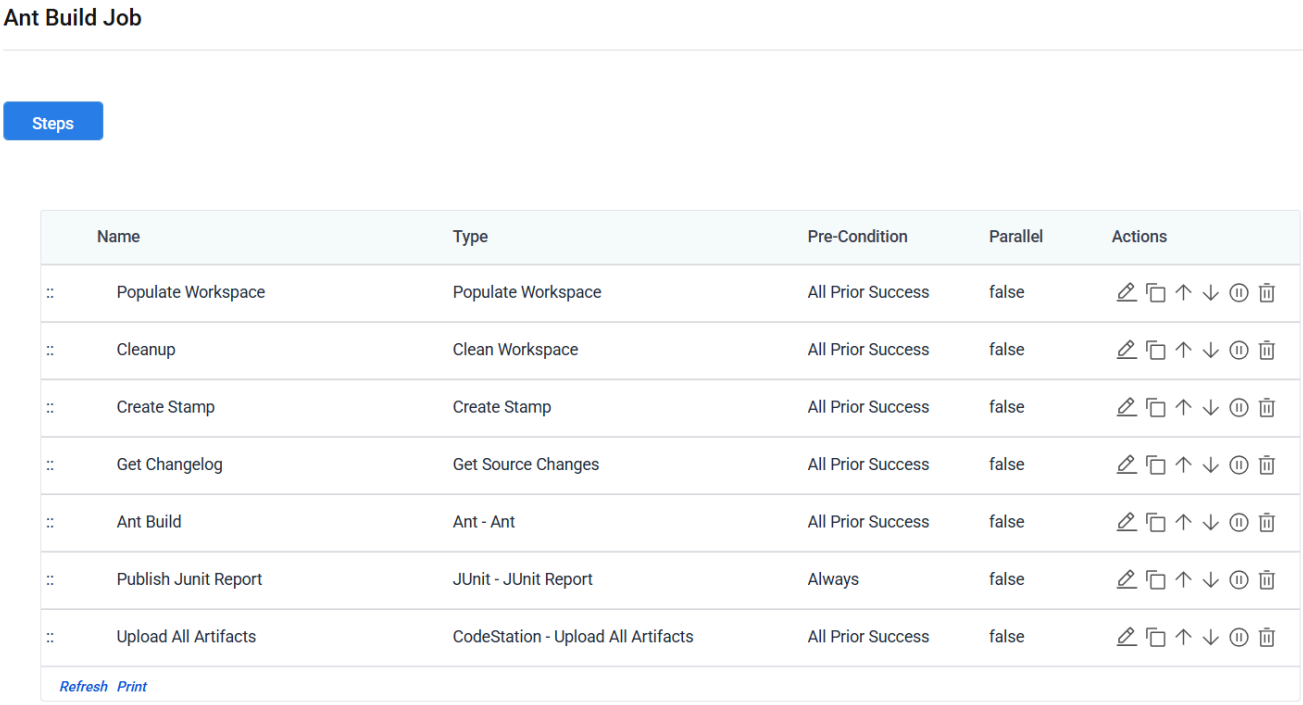
Creating the upload artifacts step

Create a step that uploads the build artifacts into the artifact repository, CodeStation.

1. Click the **Insert After** action for the Publish JUnit Report step.
2. In the **Steps** dialog box, click **Build Systems > CodeStation > >Upload All Artifacts**, and then click **Select**.
3. In the **Name** field, type `Upload All Artifacts`.
4. Accept the default values for the other fields and click **Save**.

Results

The Upload All Artifacts step is inserted into the job after the Publish JUnit Report step, as shown in this figure:



Creating the step to upload artifacts to DevOps Deploy

Create a step to upload the build artifacts to HCL DevOps Deploy (Deploy).

Before you begin

Make sure that you created an integration with Deploy, and created a component that is named Ant as described in Configure the tutorial system on page 270.

About this task

This step creates a component version and loads the build artifacts into the Deploy Ant component that you created when you configured the tutorial system.

1. Click the **Insert After** action for the Upload All Artifacts step.
2. In the **Steps** dialog box, click **Deploy > Tools > DevOps Deploy > Upload All Artifact Sets**, and then click **Select**.
3. In the **Name** field, type `Upload All Artifact Sets to UCD`.
4. In the **Component Name** field, type `${p:project/name}`.

   Note: This property value loads the artifacts into an Deploy component with the same name as the project that runs the build. Later you create a tutorial project that is named Ant. If you name your project something else, make sure that you name a component in Deploy with the name that you use for the project.

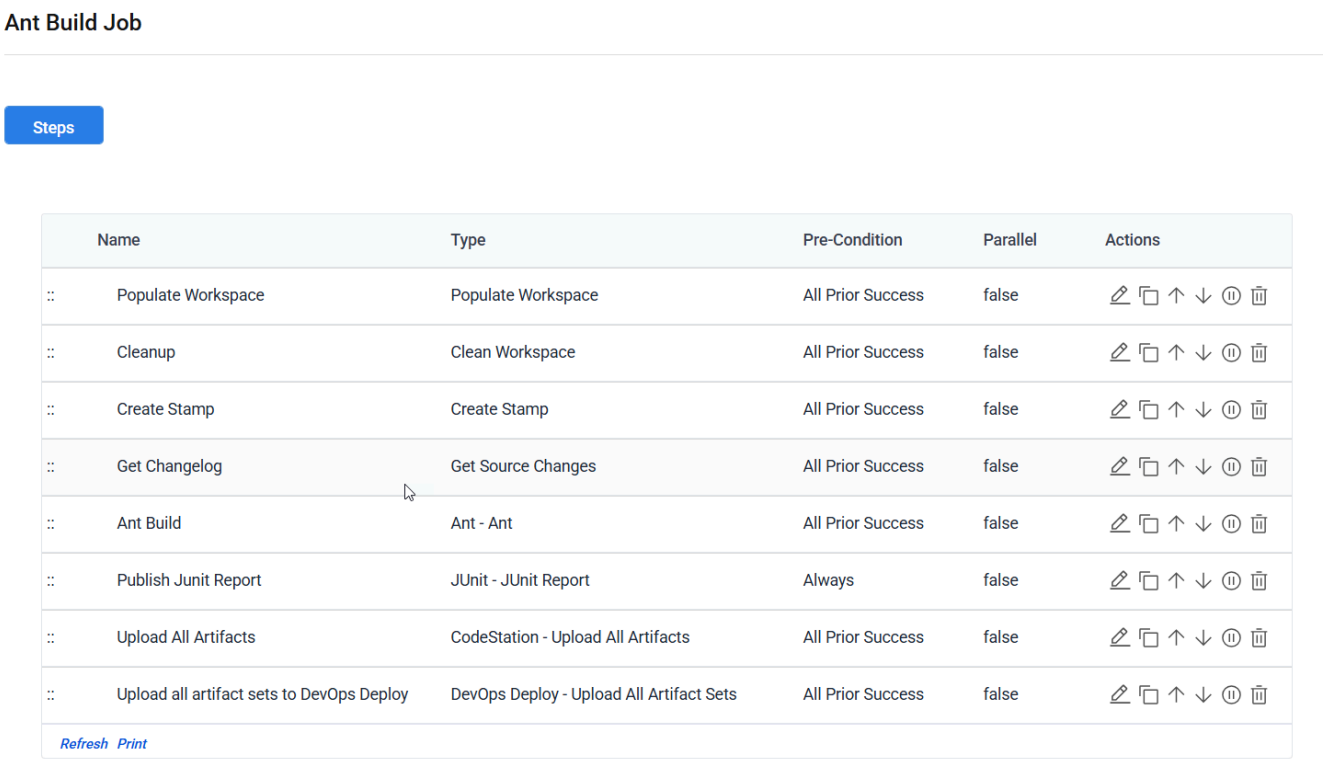
5. In the **Component Version** field, type `${p:buildlife/latestStamp}`.
   This value generates a version number for the Deploy component version. The component version ID that is created in Deploy is equal to the value that the Create Stamp job step creates.

6. In the **DevOps Deploy Server** list, select the server that you integrated with Build.
7. Accept the default values for the other fields and click **Save**.

   Result

Results

The Upload All Artifact Sets to UCD step is inserted into the job after the Upload All Artifacts step, as shown in this figure:



Creating the component version status step

Create a step that sets the component version status in HCL DevOps Deploy (Deploy).

Before you begin

Before you begin this lesson, make sure that you created a component version status with a value of `Success` in Deploy, as described in Configure the tutorial system on page 270.

About this task

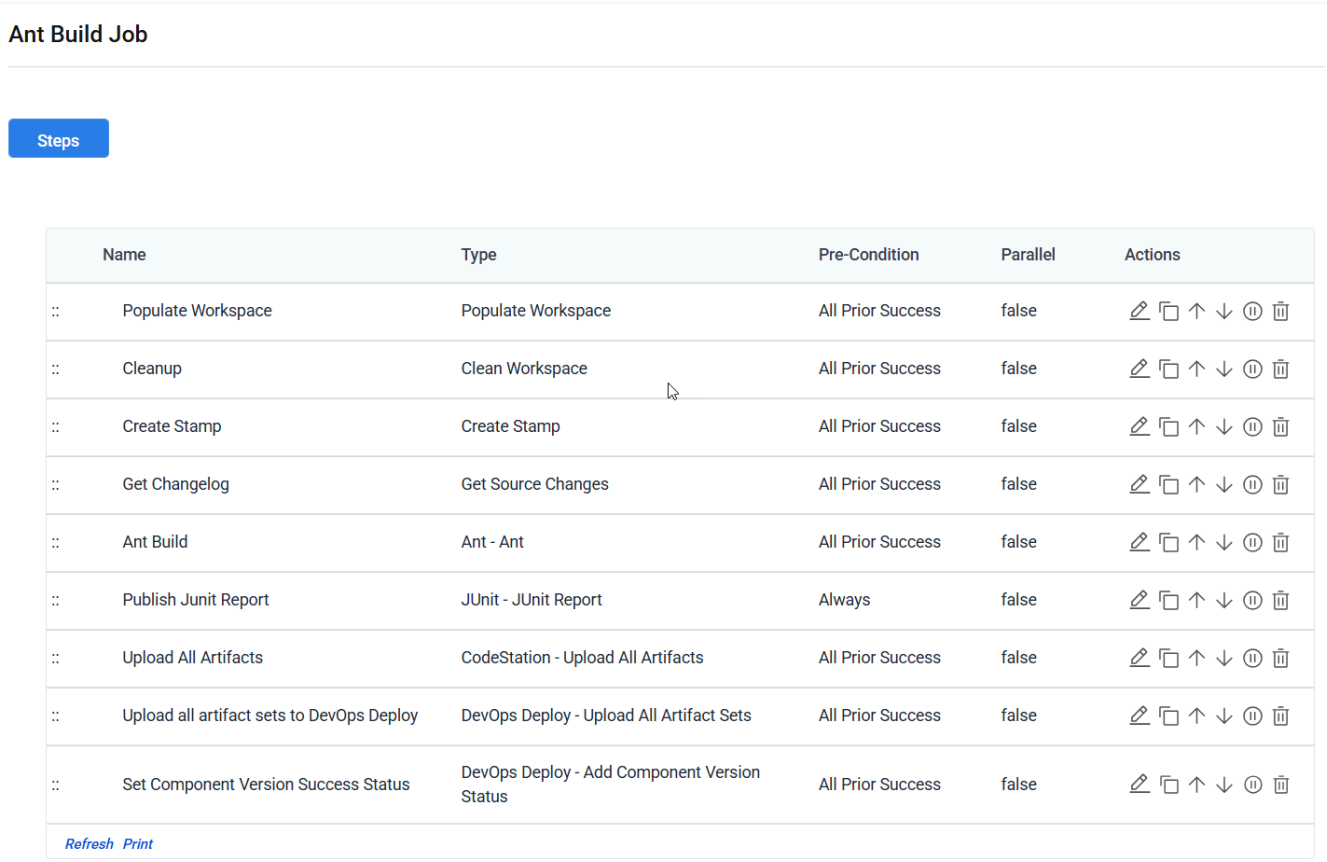
This step adds a status to the component version that was created in Deploy in the previous step.

1. Click the **Insert After** action for the Upload All Artifact Sets to UCD step.
2. On the **Steps** dialog box, click **Deploy > Tools > DevOps Deploy > Add Component Version Status**, and then click **Select**.
3. In the **Name** field, type `Set Component Version Success Status`.
4. In the **Component Name** field, type `${p:project/name}`.
5. In the **Version Name** field, type `${p:buildlife/latestStamp}`.

6. In the **Status Name** field, type `Success`.
   The status value is applied to the component version that was created in the previous step.
7. In the **DevOps Deploy Server** list, select the server that you integrated with Build.
8. Accept the default values for the other fields and click **Save**.

Results

The Set Component Version Success Status step is inserted into the job after the Upload All Artifact Sets to UCD step, as shown in this figure:



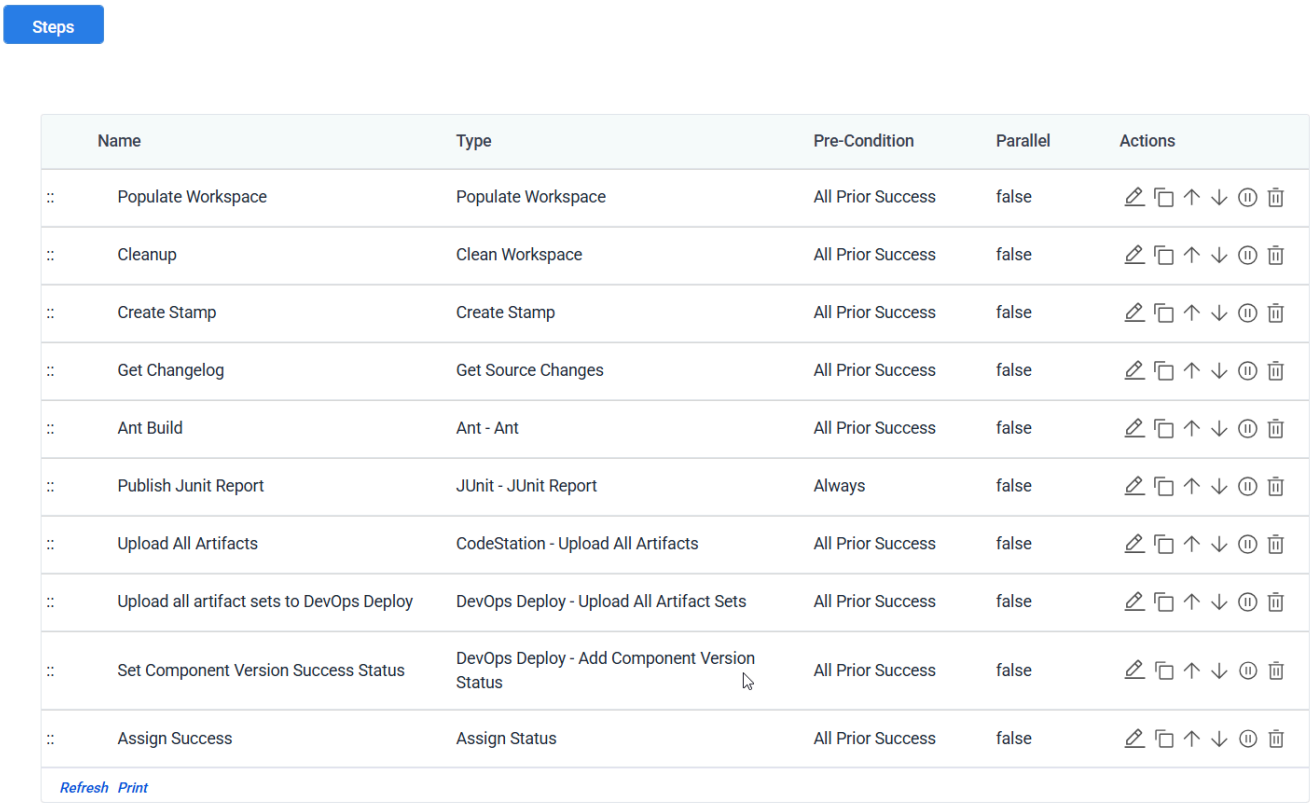
Creating the assign success step

If all previous steps succeed, this step assigns a status of Success to the entire job. If the preceding steps fail, this step does not run, and the next step runs instead.

1. Click the **Insert After** action for the Set Component Version Success Status step.
2. In the **Steps** dialog box, click **Build**, and then click **Assign Status**.
3. In the **Name** field, type `Assign Success`.
4. In the **Status** list, select **Success**.
5. Click **Save**.
   Result
   Because the **Pre-Condition Script** field is set to **All Prior Success** by default, you can accept the value.

Results

The Assign Success step is inserted into the job after the Set Component Version Success Status step, as shown in this figure:



Creating the assign failure step

This step assigns a status of Failure to the job. The step runs if any of the preceding steps fail.

1. Click the **Insert After** action for the Assign Success step.
2. In the **Steps** dialog box, click **Build**, and then click **Assign Status**.
3. In the **Name** field, type `Assign Failure`.
4. In the **Status** list, select **Failure**.
5. Display the **Pre-Condition Script** list, and select **Any Prior Failure**.
6. Click **Save**.

Results

The Assign Failure step is inserted into the job after the Assign Success step.

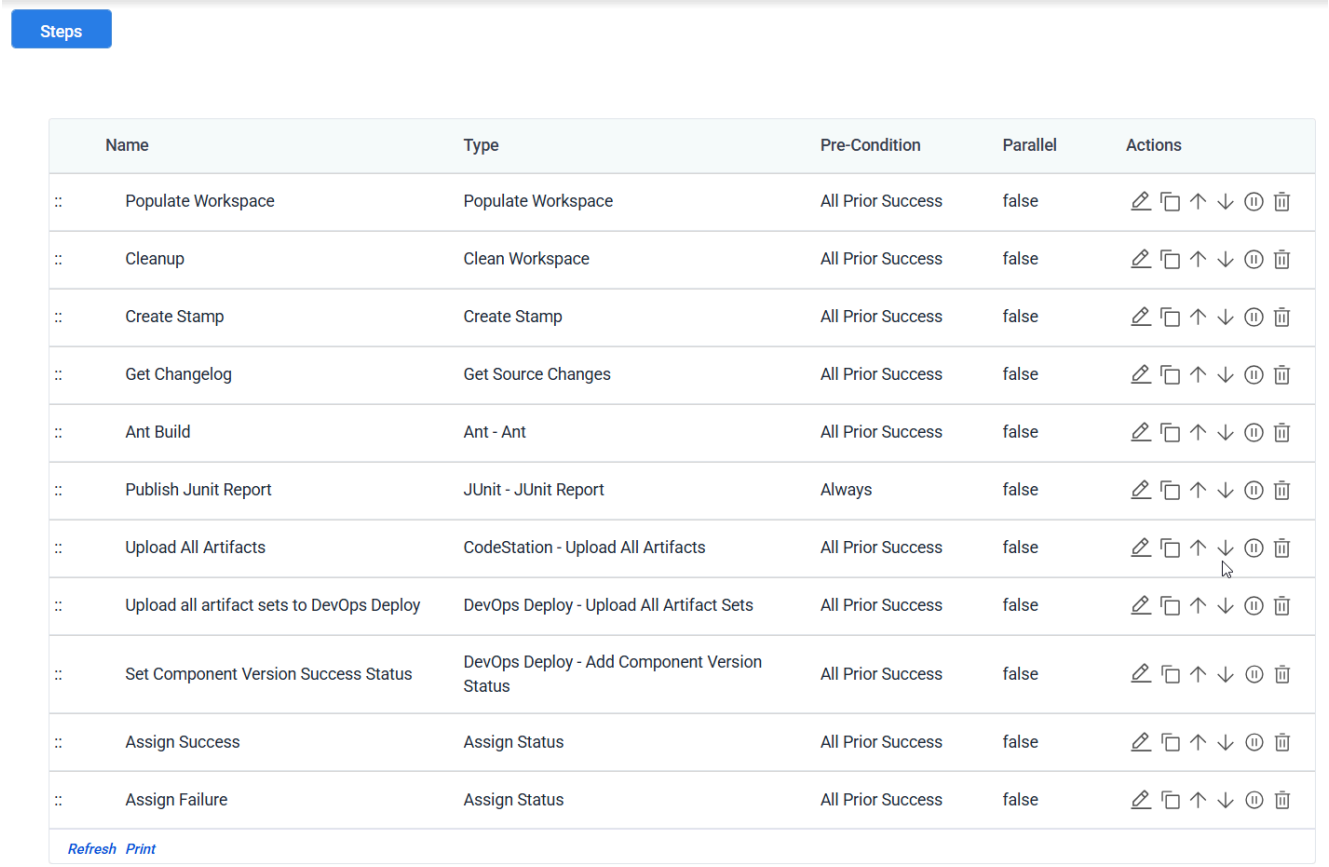
Lesson checkpoint

In this lesson, you created job and added steps to it.

About this task

This job retrieves source from GitHub and runs an Ant build process on it. It uploads the artifacts into HCL DevOps Deploy (Deploy) and generates JUnit test reports. Steps are configured individually as you insert them into a job.

The steps for the completed Ant Build Job are shown in this figure:



Lesson 3: Creating a source template

A source template is a reusable source code configuration that defines a source-code build type. Source templates can be used by any project that requires the same source type.

Before you begin

Ensure that you installed the Git source-type plug-in before you start this lesson.

About this task

In this lesson, you define a Git source configuration that the tutorial project uses.

1. On the left navigational panel, click **Source**, and then click **Create**.
2. In the **New Source Config Template** dialog, in the **Source Type** list, select **Git**.
   All installed source-type plug-ins are listed.
3. Click **Select**.
   Result
   Fields that are related to the selected source type are displayed.

4. In the **Name** field, type `gitHubSource`.
5. In the **Working Directory Script** list, select **Default Workflow Directory**.
6. In the **Remote URL** field, type `${p:remote.url}`.
   This value refers to a property that you create in the next section. You supply the property value when you configure the project.
7. In the **Remote Name** field, type `${p:remote.name}`.
8. In the **Branch** field, type `${p:branch}`.
9. Click **Save**.

Results

The **gitHubSource** source template is listed along with the other source templates on the **Source Templates** tab.

Creating properties for the source template

Define the properties that you referred to in the previous section.

1. If the **Source** tab is not already displayed, then on the Build dashboard, click **Templates**, click **Source**, and then click the **gitHubSource** source template that you created earlier in the lesson.
2. Click **Properties**, and then click **Create**.
3. Create the `remote.url` property by completing these steps.
   a. In the **Name** field, type `remote.url`.
   b. In the **Label** field, type `Remote URL`.
      The label text identifies the property to users who use the source template.
   c. In the **Display Type** list, select **Text**.
   d. In the **Value** area, accept the default option, **Defined**.
      The **Value** option determines when and where the value for the property is defined. By selecting **Defined**, you can provide a value in the **Default Value** field, or supply the value on the project configuration before the project runs.
   e. Select the **Value Required** check box.
   f. Leave the **Default Value** field blank.
      You define the value when you configure the project.
   g. Click **Save**.
4. Create the `remote.name` property by completing these steps.
   a. In the **Properties** tab, click **Create**.
   b. In the **Name** field, type `remote.name`.
   c. In the **Label** field, type `Remote Name`.
   d. In the **Display Type** list, select **Text**.
   e. In the **Value** area, accept the default option **Defined**.

f. Select the **Value Required** check box.
g. In the **Default Value** field, type `origin`.
h. Click **Save**.

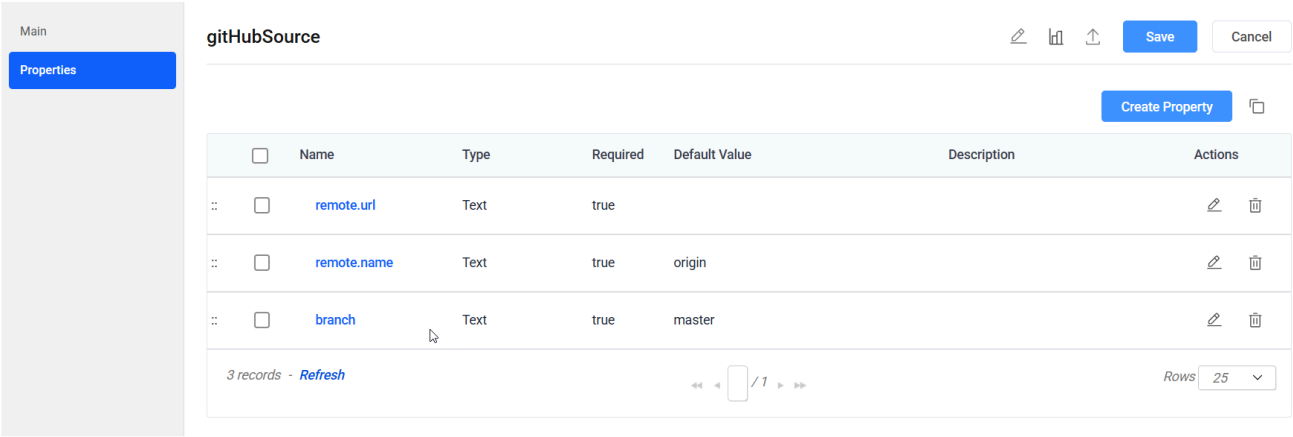
5. Create the `branch` property by completing these steps.
   a. In the **Properties** tab, click **Create**.
   b. In the **Name** field, type `branch`.
   c. In the **Label** field, type `Branch`.
   d. In the **Display Type** list, select **Text**.
   e. In the **Value** area, accept the default option **Defined**.
   f. Select the **Required** check box.
   g. In the **Default Value** field, type `master`.
   h. Click **Save**.

Lesson checkpoint

In this lesson, you created a source template and defined properties that identify the source material that is used in the build.

About this task

The gitHubSource source template properties are shown in this figure:



Lesson 4: Creating a process template

Build process templates associate processes with project templates.

About this task

In this lesson, you create a process template and assign the workflow that you created to it. In addition, you define several project properties and the artifact set that the project uses.

1. On the left navigational panel, click **Templates**, click **Process**, and then click **Create**.
2. In the **New Process Template** dialog box, in the **Name** field, type `Ant Build Process`.
3. Under **Type**, make sure that **Build Process** is selected.
4. Click **Save**.

Results

The **Main** tab for the new process template is displayed.

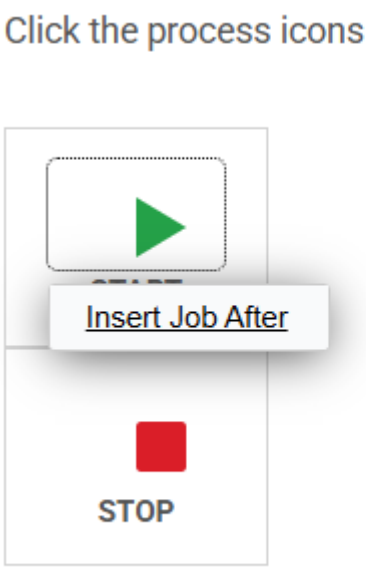
Creating a process definition

Process definitions identify the jobs in a build.

About this task

Process definitions define the jobs and steps in a build. When you place a job in a process definition, you specify several options, such as the agent pool and the working directory. Process definitions can have several jobs attached to them and, like the steps in a job, you control their order. Process definitions are created with reusable templates. In this lesson, you create a process definition and assign the job that you created in the previous lesson to it.

1. On the Build dashboard, click **Templates > Process**.
2. In the **Process Templates** page, click **Ant Build Process**, and then click **Definition**.
3. In the **Definition** area, click **Start**, and then select **Insert Job After**.
   The **Insert Job After** action is displayed after you click **Start**, as shown in this figure:



4. In the **Insert Start Job** window, in the **Job** list, select the job that you created in a previous lesson, **Ant Build Job**.
5. In the **Pre-condition** list, select **Always**.
6. In the **Agent Selection** area, select **Agent Pool Selection**, and then select the agent pool that you created when you configured your tutorial system on page 270.
   An agent in the agent pool runs the jobs that are assigned to the process definition.
7. In the **Working Directory** list, select **Default Workflow Directory**.
   An agent's default working directory is as follows:

   ```
   agent_directory/var/work/project/project_name
   ```

When this option is selected, the agent creates a folder for each project that it runs.

8. Accept the default values for the other fields and click **Insert Job**.
   Result
   The **Ant Build Job** is inserted in the **Ant Build** process definition.

Creating process template properties

Define the property that you referred to in Lesson 2.

About this task

In this step, you define properties that every project that uses the template can use. Because this template is related to Ant builds, most properties are related to Ant builds. Some properties have default values defined for them, while others are left to be defined later.

1. Click **Properties**, and then click **Create**.
   Use the values in the following table to create the properties for the tutorial workflow. You can enter any text for the property labels but remember that the labels are displayed to users who use the template. In addition, if you use the suggested label text, it is easier to follow the tutorial in Lesson 6 on page 292. If a cell is empty in the Default value column, leave the corresponding **Default Value** field blank. Accept the default values for the other fields. The descriptions you enter are displayed on some of the project's dialog boxes. Descriptions for some of the properties are provided in the figure below.

Table 7. Tutorial process template properties

Name	Label	User May Override	Type	Required	Default Value
`dir.offset`	Directory Offset	No	Text	False	
`build.script.name`	Build Script	No	Text	True	`build.xml`
`ant.targets`	Ant Targets	Yes	Text	False	`${p?:default.ant.targets}`
`build.properties`	Build Properties	Yes	Text Area	False	
`ant.properties`	Ant Properties	Yes	Text Area	False	
`product.version`	Product Version	No	Text	True	`1`
`artifacts.base.dir`	Artifacts Base Dir	No	Text	True	`.`
`artifacts.includes`	Include Artifacts	No	Text Area	True	`*`
`artifacts.excludes`	Exclude Artifacts	No	Text Area	False	

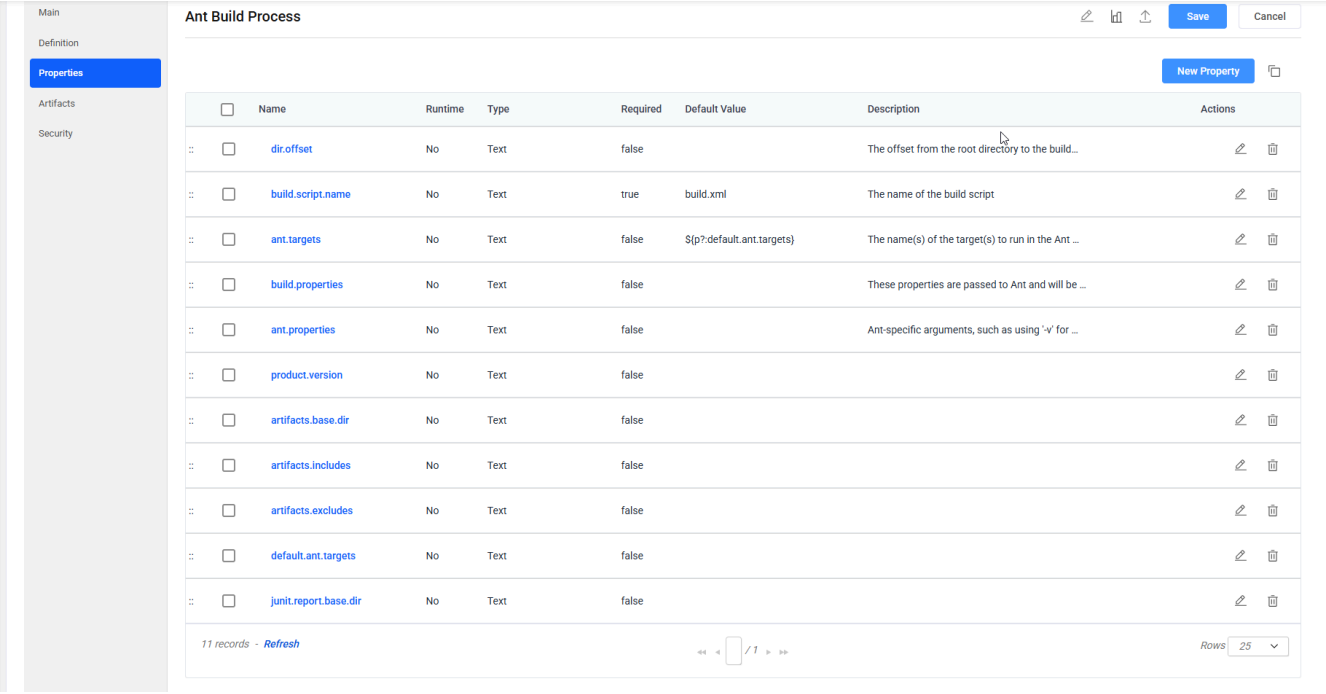
Name	Label	User May Override	Type	Required	Default Value
`default.ant.targets`	Default Ant Targets	No	Text	False	
`junit.report.base.dir`	JUnit Report Base Dir	No	Text	True	

The artifact set you create in the next section uses three properties: `artifacts.base.dir`, `artifacts.includes`, `artifacts.excludes`.

2. Click **Save** after you finish each property.

Results

The properties are listed on the **Properties** tab of the Ant Build Process process template.



Creating process artifact sets

Generated artifacts are assigned to an artifact set. An artifact set is a collection of build artifacts.

Before you begin

Before you begin this lesson, make sure that you created the artifact set as described in Configure the tutorial system on page 270.

About this task

In this lesson, you configure the artifact set the project uses. The items in the artifact set are loaded into HCL DevOps Deploy (Deploy) as a component version.

1. Ensure that the window for your build process is displayed, click **Artifacts**, and then click **New Artifact Config**.
2. In the **Artifact Set** list, select the artifact set you created earlier on page 270, **dist**.
3. In the **Base Directory** field, type `${p:artifacts.base.dir}`.
   This property and the next two properties refer to the properties you created in the last section. The default value,which was defined in the last section, means the process will use the current directory.
4. In the **Include Artifacts** field, type `${p:artifacts.includes}`.
   The default value,which was defined in the last section, means the process will include all artifacts.
5. In the **Exclude Artifacts** field, type `${p?:artifacts.excludes}`.
6. Click **Save**.

Lesson checkpoint

In this lesson, you created a process template and defined a process for it. You created properties and added an artifact set to the template.

Lesson 5: Creating a project template

Projects bring together build processes, source configurations, dependencies, and artifacts. A project template is a reusable blueprint that defines a generic project.

About this task

Because project templates are reusable, you can apply the same specifications to similar projects. Properties can be defined and the values can be supplied later in the projects that are created from the template. A project template might always use a particular type of build script, for example, but the name or location of the script can vary by project.

In this lesson, you create a project template that you use later to create the tutorial project.

1. On the left navigational panel, click **Templates**.
   The **Projects** tab is open by default.
2. Click **Create**.
3. In the **Create a Project Template** dialog box, in the **Name** field, type `Git Project Template`.
4. Click **Save**.
   Result
   The **Build Pre-Processing** tab for the new project template is displayed. You can use this tab to check for source changes or define other things to occur before the build starts.
5. Click **Agent Pool Selection**, and then select the agent pool that you created when you configured the tutorial system on page 270.
   The agent pool that you selected in Lesson 4 on page 287 will run the tutorial process. The agent pool you select here will be available during project quiet periods. A quiet period is a specified period of time during which changes cannot be committed to the source code.
6. Accept the default values for the other fields and click **Save**.

Lesson checkpoint

In this lesson, you created the project template that you use to build the tutorial project.

About this task

For more information about project templates, see Creating project templates on page 384.

Lesson 6: Creating a project

You use projects to configure and run builds. After a build runs, projects provide a wealth of information that you can use to manage the project lifecycle and troubleshoot problems.

Before you begin

Ensure that you completed all the preceding steps before you start this lesson.

About this task

Projects are built from templates. In this lesson, you use the templates that you made earlier to create the tutorial project. To define the project, you select the process and process definition templates and provide values for the properties that are defined by them.

You can run builds automatically on a schedule, or you can run builds manually, as you do in the tutorial. When you start the build, the job that you created with the job template is started. The project determines which source code to use and manages dependencies. Each build generates a unique build life

As the project runs, the build life provides feedback about the build's status and generates logs that you can use to troubleshoot problems. If projects are configured to run tests, as the tutorial project is, test results are available on the **Tests** tab for the build life.

1. On the left navigational panel, click **Projects**, and then click **Create**.
2. Select the project template that you created in Lesson 5 on page 291, **Git Project Template**.
3. In the **Name** field, type `Ant`.

   Note: The build uses the project's name to identify the HCL DevOps Deploy (Deploy) component. If you use another name for your project, make sure there is a component with the same name in Deploy.

4. Click **Save**.
   The **Configuration** tab for the project is displayed. You use this page to configure projects. You can also run project builds on this page. Later, you will use this page to run the tutorial project.

Configuring the project build process

A project must be associated with a build process. A project can be associated with any number of build processes.

1. Ensure that the **Configuration** tab for the Ant project is displayed; then, under **Build Processes**, click **New**.
2. In the **New Process** window, in the **Template** list, select **Ant Build Process**, and then click **Select**.
   The fields that you see depend on the selected template. In this case, several fields are derived from the process template that you created the previous lesson.

Note: If the only template you have is the one you created earlier in the tutorial, the template is automatically selected and this step is skipped.

3. In the **Name** field, type `GitHub Build Process`.
4. In the **Build Script** field, accept the `build.xml` value.
   You defined the default value, `build.xml`, when you created the process template.
5. In the **Product Version** field, accept the `1` value.
   You defined the default value when you created the process template.
6. In the **Artifacts Base Dir** field, type `dist/lib`.
   This overrides the default value you defined earlier.
7. In the **Include Artifacts** field, type `*.jar`.
   This overrides the default value you defined earlier.
8. Leave the **Exclude Artifacts** field blank.
9. In the **Default Ant Targets** field, type `main junit-batch`.
   You can use another target, `main junit-batch-with-failure` to automatically fail the Ant build step and produce a lot of JUnit test results.
10. In the **JUnit Report Base Dir** field, type, `build/testcases/xml`.
11. Click **Save**.

Results

The **Configuration** tab for the new build process is displayed.

Configuring the project source

You define a project's build source by assigning a source template to a build configuration and defining values for its properties.

Before you begin

Ensure that the **Configuration** tab for the build process that you defined in the previous section is displayed.

About this task

You define the source configuration by selecting the source template that you created earlier, and by using template properties to identify the source location.

1. Under **Source Configuration**, click **Create**.
2. In the **New Source from Template** dialog box, in the **Source Template** list, select the source template that you created earlier, **gitHubSource**, and then click **Select**.

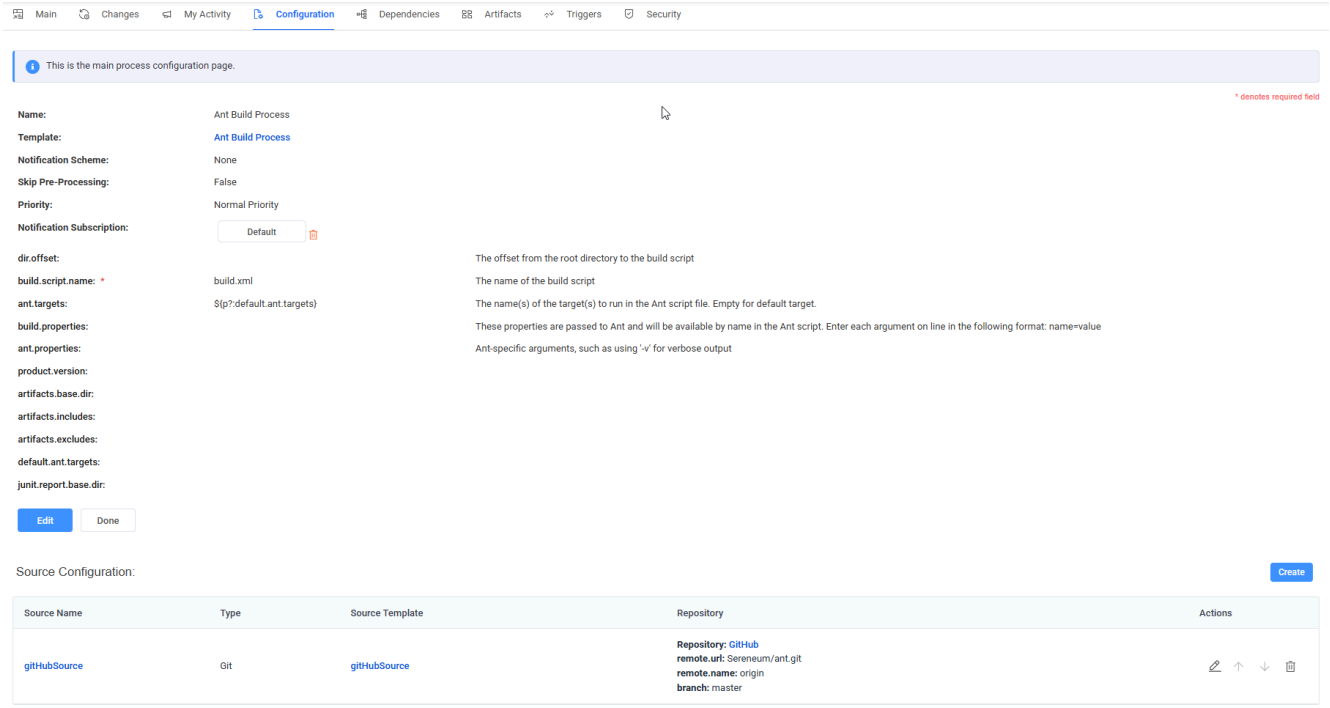
Note: If the only template you have is the one you created earlier in the tutorial, the template is automatically selected and this step is skipped.

3. In the **Name** field, type `gitHubSource`.

4. In the **Repository** list, select the repository that you created earlier, **GitHub**.
5. In the **Remote URL** field, type `Serneum/ant.git`.
   You defined this field and the next two fields when you created the source template.
6. In the **Remote Name** field, accept the default value `origin`.
   You defined the default value for this field when you created the source template.
7. In the **Branch** field, type `ANT_18_BRANCH`.
   This overrides the default value, `master`, you defined earlier.
8. Click **Save**.

Results

With the build process defined and the build source identified, the project is complete. The finished project is shown in this figure:



Lesson checkpoint

In this lesson, you created a project and defined a build process and a build source.

About this task

The project is constructed from the templates that you made earlier in the tutorial. All projects are built from templates. When you create a project, you identify the templates and provide property values for them. With this lesson finished, it's time to run the project.

Lesson 7: Running the build

Run the project build process to transform the source code into a distributable component and load it into HCL DevOps Deploy (Deploy).

Before you begin

To see the finished build objects that are loaded into Deploy, make sure that you complete these tasks before you run the build:

- Configure an instance of Deploy and the Build agent that you use in the tutorial project can access it.
- Create an Deploy integration that identifies the Deploy server that you are using in the tutorial.
- Make sure that your instance of Deploy is running
- Create a component named Ant in Deploy.
- Create a version status in Deploy that is named Success.

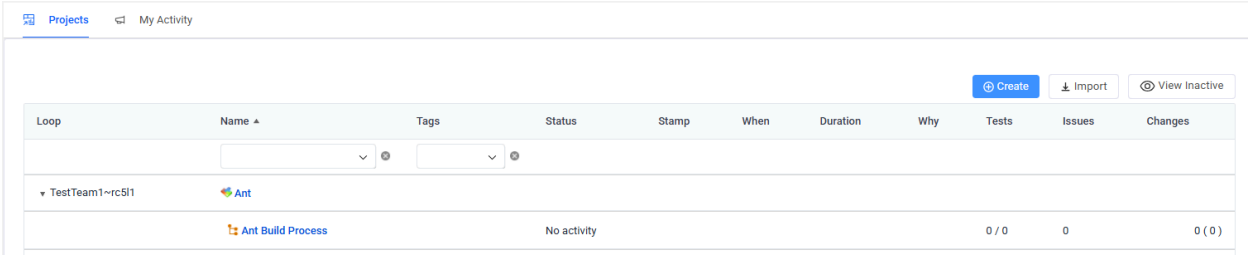
About this task

You run a build by running a build process that is defined for a project. The build process takes the source material and transforms it into the final output by using a job that is assigned to the project. The job in the tutorial project completes these activities:

1. Checks out source material from the GitHub repository.
2. Uses Ant to build the project artifacts.
3. Runs JUnit tests on the build process.
4. Loads the artifacts into an Deploy component.
5. Assigns a status of Success to the component in Deploy.

Typically when you run a build, you can use several run-time properties. The number of properties depends on the template. In the case of the tutorial project, the **Ant Targets** property affects the build's outcome. If you accept the `${p:?default.ant.targets}` default value for the **Ant Targets** property, the build process uses the `main junit-batch` value that you defined in the previous lesson. You can overwrite the default value with one of your own, but for the tutorial accept the default value. Another target is discussed later in this lesson.

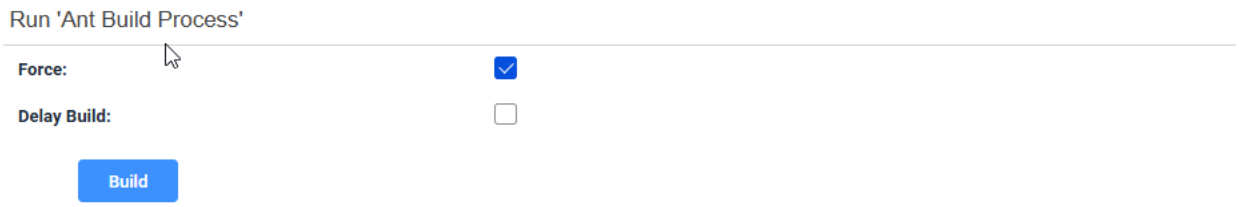
1. On the left navigational panel, click **Projects**, and then expand the project that you created in the previous lesson, **Ant**.
   Result
   Build processes that are configured for the selected project are listed beneath the project, as shown in this figure:



2. Click **Ant Build Process**.

Result

The **Main** tab for the Ant Build Process is displayed.

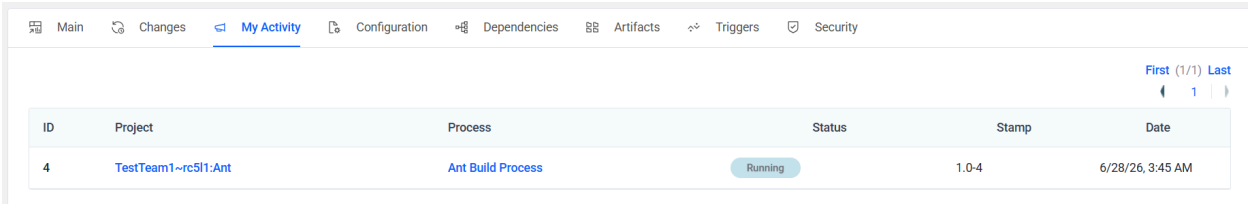


A project build process can have any number of run-time properties. The user who runs the build can enter values that are related to a particular build or override default values.

3. Accept the default value in the **Ant Targets** field.
4. Click **Build**.

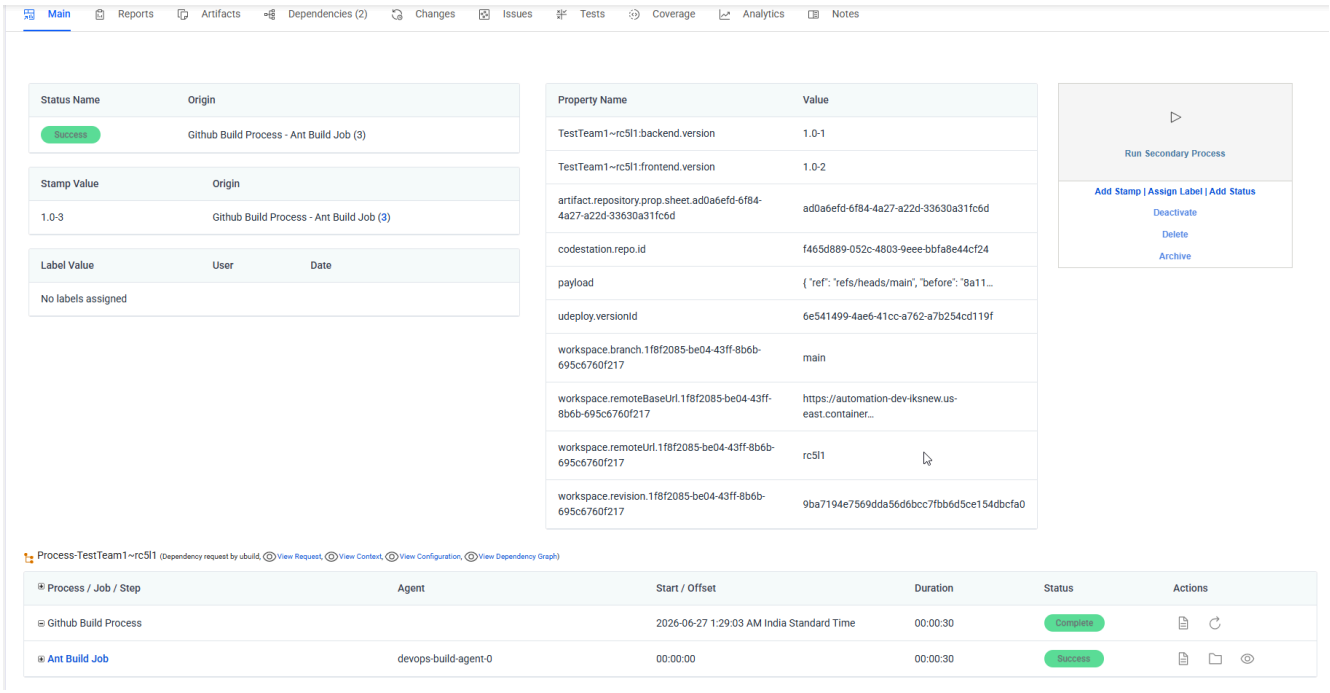
   Result

   The build process starts and the **My Activity** tab for the build is displayed. The build, with a new build life number, is listed with a status of `Running`, as shown in this figure:



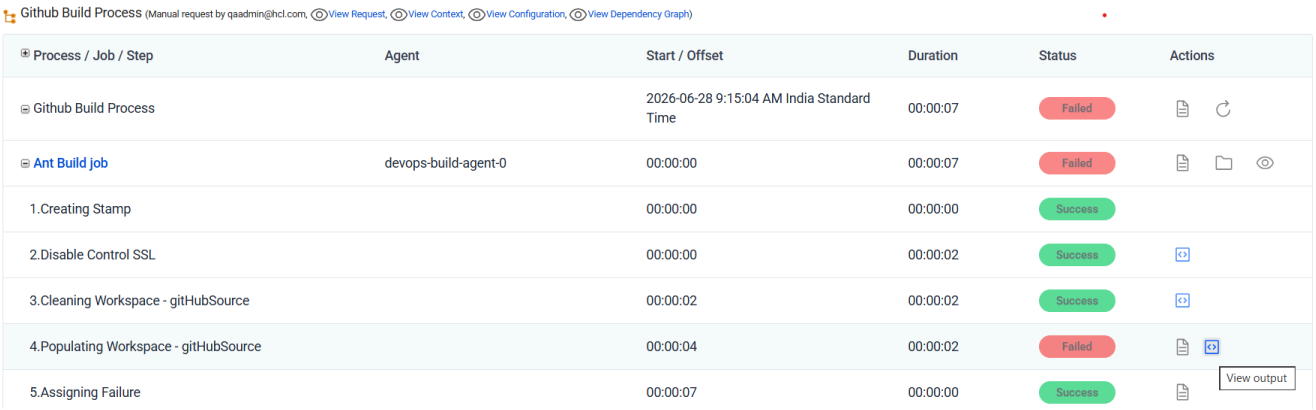
Results

Refresh the **My Activity** page. If the build succeeds, the status changes to `Complete`; otherwise, the status changes to `Failed`. The project's **Main** tab contains information about the builds that the project generated. To see information about a build, click the **ID** for the build life. A completed build of the tutorial project is shown in this figure:



What to do next

If a build fails, you can see information about a failed step. To see a step's output, use the **View Output** action, as shown in the following figure. Use the log to identify errors and troubleshoot failed steps.



The tutorial project is configured to run JUnit tests. To see the types of reports that are created when a build fails, run the build again, but use the `main junit-batch-with-failures` target in the **Ant Targets** field. This target automatically fails on the Ant Build step in the Ant Build Job, and generates numerous JUnit reports.

To see the JUnit test results, open the **Tests** tab for the build.

Lesson checkpoint

In this lesson, you ran a build of the tutorial project and examined its progress, tests, and final output.

About this task

In this lesson, you ran a build for the project that you created in the tutorial. The build created artifacts that were loaded into an Deploy component version and assigned a status of Success. Finally, the build created JUnit reports that are available on the **Tests** tab of the build life.

Lesson 8: Verifying the build results

Check the component version that was created by the tutorial project in HCL DevOps Deploy (Deploy).

Before you begin

Run the project that is described in the previous lesson and ensure the build life status is `Complete`.

About this task

The build that you ran in the previous lesson loaded artifacts into an Deploy component version and assigned a status of success to it. In this lesson, you review the component version in Deploy.

1. Access the instance of Deploy that you configured for the tutorial project.
2. From the Deploy Dashboard, click **Components**, and then click the **Ant** component you created for the tutorial.
3. Click **Versions**.

   Result

   The component version that the tutorial project created is listed on the **Versions** tab. The component version status is listed in the **Statuses** column. The tutorial project assigns `Success` status to the component version.

   The version number corresponds to the stamp that you created in the Ant Build Job.

What to do next

The artifacts that the tutorial project created are loaded in Deploy. To see the actual artifacts, expand the `dist` artifact set that you defined in the project.

Lesson checkpoint

In this lesson, you reviewed the component version that the tutorial project created in Deploy.

Summary

In this tutorial, you learned how to build a project and run a build with Build.

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How to create a repository and use it in a build process
- How to create jobs and add steps to them
- How to create and use workflow templates
- How to create source and process templates
- How to create project templates
- How to create projects and configure their sources, processes, and workflows
- How to run a build and examine its output

Administering

Users with the correct permissions can configure security, create users, configure integrations, and create notifications and announcements among other things.

Feature flags

Feature flags allow you to enable and disable product functions.

To modify the value of a feature flag, edit the `installed.properties` file and locate the feature flag. If not found, a line with the feature and its value. The format is *`featureflag_name=value`*, where the value is true or false. In the following example, the first line enables the automatic keystore generation feature and the second line disables the Build client download link from the tools menu.

```
automatic.keystore.generation.enabled=true
ucb.client.download.enabled=false
```

The following table contains a description of the available feature flags.

Note: Several feature flags control optional and experimental features.

Feature name	Description	Default Value
ucb.licensing.lazyInit	Initialize licensing in the background to avoid blocking server startup for checking licenses.	false
user.profile.locale.enabled	Allow Build users to select a locale in their profile.	true
user.profile.collation.enabled	Allow Build users to select a collation in their profile.	true
token.licensing.enabled	Allow the use of tokens for fallback licensing.	true
ucb.client.download.enabled	Show the Build client on the **Tools** page.	true
globalization.test.languages.enabled	Allow the use of special pseudo-translations for globalization testing.	false
process.dependency.graph.enabled	Enable the **Visual Dependency Graph View** button on the **Process Configuration** page.	true
oracle.nls.settings.enabled	Configure the national language support (NLS) settings for Oracle	false

Feature name	Description	Default Value
	databases. Enabling this feature might degrade the execution performance of various SQL statements in exchange for correctly sorting result sets for non-English locales.	
automatic.server.upgrade.enabled	Allow the server to run upgrade scripts during startup operations. This feature is useful when you run Build in a container. It is not applicable to non-container installations.	false
automatic.keystore.generation.enabled	Allows the Build server to generate the required keystore files and keys and certificates that are based on information from the `installed.properties` file, if the specified keystore files do not exist.	false
manage.all.roles.enabled property	Allow users with the Team User Management role to add users to all roles within the team, with the exception of the Admin role.	false

Running agents

To start an agent, run the shell script or batch file. An agent can be started before or after Build is started. The agent must be started and configured before it can be used.

1. Go to the script. *`agent_installation`*`\bin` directory.
2. Run the appropriate command.
    - On Windows, run this command:

      ```
      agent start
      ```

    - On Linux or UNIX, , run this command:

      ```
      .\agent start
      ```

3. After the agent starts, go to the Build web application, and open the **Agents** tab.

Results

When the process is complete, the agent is listed with a status of `Online` on the **Agents** page.

Starting agents on z/OS

After an agent is installed on z/OS, it can be started.

About this task

The agent can be started manually or by using Job Control Language (JCL).

Starting agents on z/OS manually

After an agent is installed on z/OS, it can be started.

About this task

The agent can be started manually or by using Job Control Language (JCL).

1. Locate the directory where the agent is installed.
2. From a command line, issue the following command to start the agent.
   bin/agent start.

   The agent process runs in the background.

Using JCL to start the z/OS agent

About this task

The agent can be started manually or by using Job Control Language (JCL).

1. Locate the JCL script in the `ibm-devops-build-agent-install/install/zos/BULAGNT` path.
2. Verify that the agent directory is correct in the JCL script file.
3. Submit the JCL script file.

Viewing z/OS agent logs

The z/OS agent log files are stored in UTF-8 charset. To read agent log files, use an editor that supports the UTF-8 format.

About this task

To view the agent log file, issue the following command.

```
vi -W filecodeset=UTF-8,pgmcodeset=IBM-1047 agent.out
```

Configuring an external agent for Build in DevOps Loop

You can configure an external Build agent to connect with DevOps Loop installed on Kubernetes (K8s), Azure Kubernetes Service (AKS), IBM Kubernetes Service (IKS), or OpenShift Container Platform (OCP).

Before you begin

Before you begin, ensure the following prerequisites are met:

- Installed DevOps Loop on Kubernetes (K8s), Azure Kubernetes Service (AKS), IKS, or OCP.
- Ensured that you have access to the Kubernetes or OpenShift cluster where DevOps Loop is running.

- Installed the DevOps Build agent on a supported external platform.
- Been granted permissions to edit services or create routes in the cluster.

1. Edit the Emissary ingress service to expose the Build agent port for Kubernetes (K8s), Azure Kubernetes Service (AKS), or IKS deployments.

    a. Run the following command:

    ```
    kubectl edit svc emissary-ingress -n emissary
    ```

    b. Under the *spec.ports* section, add the following configuration:

    ```
    - name: build-agent
    nodePort: 31123
    port: 7920
    protocol: TCP
    targetPort: 7920
    ```

2. Update the `installed.properties` file of the agent located at `<agent-install-dir>/conf/agent` with the cluster domain and port.

    a. For Kubernetes (K8s) environments:

    ```
    locked/agent.brokerUrl=failover:(ah3://{{.Values.global.domain}}:7920)
    locked/agent.jms.remote.host={{.Values.global.domain}}
    locked/agent.jms.remote.port=7920
    ```

    `Values.global.domain` is the same URL used to access the DevOps Loop UI.

    b. For IKS environments:

    ```
    locked/agent.brokerUrl=failover:(ah3://<service-domain>:7920)
    locked/agent.jms.remote.host=<service-domain>
    locked/agent.jms.remote.port=7920
    ```

    `<service-domain>` is the domain configured for the IKS cluster in Step 4 on page 302.

3. Start the external Build agent by running the following command:

    ```
    sudo <agent-install-dir>/bin/agent run
    ```

    Result

    The agent is displayed under the **Agents** section in the Build UI.

4. For IKS, configure the cluster service domain, if it is not configured.
    Follow steps in the Installing DevOps Loop on IBM Cloud Kubernetes Service (IKS) on page 46 topic for setting up the service domain.
5. **Optional:** If the above approach does not work, use the following alternate method for a Kubernetes cluster running on a demo RHEL setup.

    a. Patch the service to expose an external IP and port.
    Run the following commands:

    ```
    kubectl patch svc devops-build-server -n devops-loop --type=merge -p
     '{"spec":{"externalIPs":["<machine-ip>"]}}'
    ```

```
kubectl patch svc devops-build-server -n devops-loop --type=json
 -p='[{"op":"replace","path":"/spec/ports/2/port","value":<required-port>}]'
```

Replace `<machine-ip>` and `<required-port>` with appropriate values. Ensure that the specified port allows inbound and outbound traffic from the host machine where the external agent is running.

b. Update the `installed.properties` file of the agent.

```
locked/agent.brokerUrl=failover:(ah3://<machine-ip>.nip.io:<required-port>)
locked/agent.jms.remote.host=<machine-ip>.nip.io
locked/agent.jms.remote.port=<required-port>
```

6. For OpenShift deployments, create a route to expose the Build agent service.

a. Apply the following route definition:

```
cat <<EOF | oc apply -n <namespace> -f -
apiVersion: route.openshift.io/v1
kind: Route
metadata:
name: devops-build-agent
spec:
to:
kind: Service
name: devops-build-server
port:
targetPort: agent
tls:
termination: passthrough
insecureEdgeTerminationPolicy: Redirect
EOF
```

b. Retrieve the route host by running the following command:

```
oc get route devops-build-agent
```

c. Update the `installed.properties` file of the agent located at `<agent-install-dir>/conf/agent` as follows:

```
locked/agent.brokerUrl=failover:(ah3://<route-host>:443)
locked/agent.jms.remote.host=<route-host>
locked/agent.jms.remote.port=443
```

d. Start the external Build agent by running the following command:

```
sudo <agent-install-dir>/bin/agent run
```

Result

The agent is displayed under the **Agents** section in the Build UI.

Results

You have configured the external Build agent and connected it to DevOps Loop.

What to do next

You can now run jobs on this agent.

Integrating Build resources with existing loop and teamspaces

You can integrate the previously created Build resources, including teams, mappings, and agent pools to an existing teamspace and loop after you upgrade from DevOps Loop 1.0.2 to 1.0.3 or later.

Before you begin

You must have completed the following tasks:

- Ensured that you have the administrator privileges in DevOps Loop.
- Ensured that you have a valid bearer token.
- Ensured that you have a loop ID, teamspace ID, and loop name.

1. Create a team in Build by using the following API call:

```
curl -kX PUT "${LOOPDOMAIN}/build/rest2/team/loop/${loopId}" \
-H "Authorization: Bearer ${BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-d '{
"name": "${LoopName}",
"description": "",
"type": "Loop",
"teamspaceId": "${teamspaceId}",
"roleMappings": [
{
"role": "Platform User",
"user": "${UserEmailAddress}",
"authenticationRealm": "Platform_build_OIDC"
}
]
}'
```

   Replace all variables with actual values:

   - **loopId**: Navigate to **Deploy > Settings > Teams**. Select the team created during loop initialization and loop ID can be found at the end of the URL.
   - **teamspaceId**: Navigate to **Deploy > Settings > Teams**. Select the team created during teamspace creation and teamspace ID can be found at the end of the URL
   - **LoopName**: Use the loop's name, without any prefix.
   - **BEARER_TOKEN**: Use a valid bearer token.

   You can assign additional members in the `roleMappings` array. Supported roles include `Platform User` and `Platform Admin`.

2. Perform the following steps to map resources to the newly created team in Build:

   a. Click **Teams** in the left navigation panel.
   The **Teams** tab is displayed with the list of teams.

   b. Select the newly created team.

c. Map the following resources in the **Team Objects Mappings** section:
   - Project
   - Project Process
   - Job
   - Repository
   - Template
   - Agent Pool

d. Select each resource from the drop-down, click **Add**, select the newly created team, and click **OK**.

3. Add the required users to the new team.
4. Configure the agent and assign it to the appropriate agent pool.
5. Click **Retry** for the failed teamspaces after upgrading from DevOps Loop 1.0.2 to DevOps Loop 1.0.3 or later to complete the teamspace creation.

Results

The Build environment is now successfully integrated into the previously created loops and teamspaces after upgrading from DevOps Loop 1.0.2 to DevOps Loop 1.0.3 or later.

Related information

Configuring Agents

Assigning users to teams

Integrations

When you configure integrations, you allow Build to use and manage the objects and artifacts that are produced in external tools such as HCL DevOps Deploy (Deploy).

Build provides integrations with many tools. Integrations are implemented at the job and step levels. With these integrations, Build can distribute process execution and access third-party tools.

Integrating with Eclipse

Eclipse projects can be integrated with Build.

Build includes a plug-in for use with Eclipse. Use this plug-in to view the current activity and state of Build projects, to run builds from Eclipse, and to map Eclipse projects to projects in Build.

Installing the Eclipse plug-in

To integrate with Eclipse, install the Eclipse plug-in that comes with Build.

1. In Build, click the **Help** icon, and then click **Tools**.
2. Copy the URL for the Eclipse plug-in.
   The URL format is `http://build_server.example.com/tools/eclipse`.
3. In Eclipse, click **Help > Install New Software**.
4. In the **Add Repository** window, specify a name for the repository, copy the URL for the Eclipse plug-in into the **Location** field, and then click **OK**.
5. Select **Build Plugins** from the list of available software, and then click **Next**.
6. Read and accept the license terms, and then click **Finish**.
7. After the installation is complete, restart Eclipse.

Configuring the Eclipse plug-in

After you install the Eclipse plug-in, you must configure the plug-in. Use the Eclipse plug-in to browse Build projects and processes and to start a build in a process.

1. In Eclipse, click **Window > Preferences > Build Preferences**.
2. Specify values for the Build settings.

Preference	Value
URL	The base URL to the Build server, including the protocol. Specify a port number if the Build server does not use the default port. For example, specify `https://server.example.com:8443`.
User	The Build user name.
Password	The Build password.
Recent Process Activities Displayed	The number of recent build lives to display for each process. The default value is 5. You can specify up to 999 recent build lives to display.

3. Click **Window > Show View > Other**, and then expand **Build**, and select **Build View**.
4. Click **OK**.
   Result
   The Build view opens. Projects are displayed in the Build Projects folder. To start a build, right-click a process, and then click **Run Process**.

Results

The Build view is available in Eclipse.

What to do next

To resolve dependencies in a project, configure CodeStation preferences.

Configuring CodeStation preferences

To resolve dependencies in a project, you must configure CodeStation preferences in the Eclipse plug-in.

Before you begin

Create a Java project in Eclipse. The project dependencies are stored in the Java project.

1. In Eclipse, click **Window > Preferences > CodeStation Preferences**.
2. Specify values for the preferences.

Preference	Value
URL	The base URL to the CodeStation server, including the protocol. Specify a port number if the CodeStation server does not use the default port. For example, enter `https://server.example.com:8443`.
User	The Build user name.
Password	The Build password.

3. In Eclipse, right-click the Java project, and then click **Build Path > Add Libraries**.
4. Select **CodeStation Managed Artifacts** from the list.
    a. Under **Build Project**, select the Build project that corresponds to the Eclipse project.
    b. Under **Build Process**, select the Build process that corresponds to the Eclipse project.
    c. To lock the dependency artifacts to a build life, enter a build life number under **Build Life ID**. If **Build Life ID** is left blank, the latest dependencies are retrieved.
5. Go to the **Console** view, click **Open Console**, and then select **CodeStation Console** from the list.
6. Go to the **Package Explorer** view, right-click the Java project, and then select **CodeStation > Resolve Dependencies**.

   Result

   The plug-in downloads dependencies from Build to the Eclipse project.

Integrating with DevOps Deploy

HCL DevOps Deploy (Deploy) applications and components can be integrated into Build.

Before you begin

You can use a user account or authentication token to connect to Deploy. To use a user account, obtain a user ID and password. To use an authentication token, use Deploy to generate a token. Ensure that the server is available and you have its URL.

1. Click **System > DevOps Deploy**.
2. On the **DevOps Deploy** page, click **Create New Integration**.
3. Specify a name for the integration in the **Name** field.
4. Type the URL for the Deploy server in the **DevOps Deploy URL** field.

   The format is `[HOST]:[PORT]`. For example, type: `http://devops_deploy.domain.com:8080`.

5. Type the user name for the Deploy account in the **Name** field.
6. Type the password for the account in the **Password** field.
7. **Optional:** To use a property to generate the password, enter the script in the **Password Script** field.
8. To use an authentication token instead of user credentials, enter or paste the authentication token Deploy generates in the **Auth Token** field.
   Authentication tokens are created with the **DevOps Deploy Token** page. (**Settings > Tokens**.)

   Note: The token must be generated for the `admin` user.

9. Select the Deploy version number from the **Version** list.
10. Click **Save**.

Results

Build job steps can create and update Deploy component versions, component version statuses, properties, and artifacts.

Integrating with Maven

Build can act as a Maven repository for projects that are built in Build and as a caching proxy for external Maven repositories.

About this task

You can configure Maven repositories and the artifacts from them can be used in CodeStation projects. Downloaded Maven projects are stored as CodeStation projects. See CodeStation projects on page 372.

1. Click **System > Maven**.
2. On the **Maven** page, click **Edit**, and then select a trigger type from the **Dependency Trigger Type** options.

   This option determines the type of build trigger that is created when project dependencies are detected. When a project builds, push triggers initiate builds for dependent parent projects. When a project builds, pull triggers initiate builds for child projects that changed since the last time the parent was built.
3. **Optional:** To configure a Maven repository, see Creating Maven repositories on page 331.
4. Click **Done**.

Managing scripts

The script library provides numerous scripts for you to use and modify.

Javadoc™ documentation on the **Tools** page describes the scripting system. Scripting objects are described and numerous examples are provided.

To access the scripting tools, click **Help > Tools > Scripting API**.

Creating job-precondition scripts

Job-precondition scripts are used to determine whether a job can run.

About this task

A typical precondition is to require that all ancestor jobs are complete before the job runs.

1. On the left navigational panel, click **System > Job Precondition** in the **Script Library** area.
2. Click **Create New Script**.
3. In the **New Job Precondition Script** window, type a name for the script in the **Name** field.
4. Enter the script in the **Script** box.
   The script can be any JavaScript™ code that results in a true or false value.
5. Click **Save**.

What to do next

To review or edit an existing script, select the script. For information about scripting, see the **Build Scripting** page: click **Help > Tools > Scripting API**.

Creating post-processing scripts

Post-processing scripts are used in job steps to determine the success or failure of a step. Success is determined by the exit code and log of the step.

About this task

A typical post-processing script might check for a nonzero exit code, or look for some expected string in the step output.

1. On the left navigational panel, click **System > Post-Processing** in the **Script Library** area.
2. Click **Create New Script**.
3. On the **New Post-Processing Script** window, type a name for the script in the **Name** field.
4. Enter the script in the **Script** box.
   The script must be written in JavaScript.™ The step output properties are in a `java.util.Properties` variable named `properties`. The `exitCode` property contains the exit code of the process. The `Status` property contains the final status. Any `Status` value other than Success results in the step failing. The `scanner` variable can be used to scan the output log of the step. The `scanner` variable has several public methods:
   - `register(String regex, function call)` registers a new function to be called whenever the regular expression is matched.
   - `addLOI(Integer lineNumber)` adds a specific line to the lines of interest list. The lines are displayed in the LogViewer after the process finishes. This is implicitly called anytime `scan()` matches a line.
   - `getLinesOfInterest()` returns a `java.util.List` list that contains the lines of interest. This method can be used to remove lines if necessary.

   After all regular expressions are registered, use `scan.scan()` to scan the log.
5. Click **Save**.

Example

```
commandOut.println("This is a test command output line! This can be used as a means of logging messages!");
var prop1Value = "";
```

```
scanner.register("^prop1=", function(lineNumber, line) {
  commandOut.println("Inside callback for line of interest: " + line);
  prop1Value = line.substr(line.indexOf('=') + 1, line.length - line.indexOf('='));
  properties.put("buildlife/prop1", prop1Value);
});
scanner.scan();
commandOut.println("Exit code is: " + properties.get("exitCode"));
commandOut.println("Value for property 1 is: " + prop1Value);
properties.put("Status", "Success");
```

The example includes a couple of statements with the commandOut command. You can use commandOut command to print messages that can be used for tracing flow and debugging.

In regards to setting Status, while there is nothing wrong with explicitly setting a Failure status, there is no need to. Lack of a Status property with a value of Success is equivalent to Failure. Basically, anything other than Success is considered Failure.

What to do next

To review or edit an existing script, select the script. For information about scripting, see the **Build Scripting** page: click **Help > Tools > Scripting API**.

Creating step-precondition scripts

Step-precondition scripts define the conditions that must be fulfilled before a step can run.

About this task

A typical step precondition might require that all prior steps are complete before the step can run.

1. On the left navigational panel, click **System > Step Precondition** in the **Script Library** area.
2. Click **Create New Script**.
3. On the **New Step Precondition Script** window, type a name for the script in the **Name** field.
4. Enter the script in the **Script** box.
   The script can be any JavaScript™ code that results in a true or false value.
5. Click **Save**.

What to do next

To review or edit an existing script, select the script. For information about scripting, see the **Build Scripting** page: click **Help > Tools > Scripting API**.

Creating working directory scripts

Project source configurations use working directory scripts to determine where checked-out source files are placed.

About this task

Working directory scripts can contain hardcoded paths or literal values. Every plug-in has access to working directory scripts.

1. On the left navigational panel, click **System > Work Dir** in the **Script Library** area.
2. Click **Create New Script**.
3. Type a name for the script in the **Name** field.

4. Enter the path to the working directory in the **Script** box.
   The script can include `$` and `{}` variables, and `${bsh:...}` scriptlets.
5. Click **Save**.

What to do next

To review or edit an existing script, select the script. For information about scripting, see the **Build Scripting** page: click **Help > Tools > Scripting API**.

User notifications

Build can send information about build results to selected users by email, instant messaging, and other methods.

Notifications are managed by notification schemes. A notification scheme defines the rules that determine who receives notices, and the events that trigger notification. Processes are configured with notification schemes, and a single scheme can be shared by other processes.

Notifications can be sent to users in various formats. Most commonly, notifications are configured as email or IM messages. Recipients can be non-users as well as users.

Configuring mail servers

Build uses an external mail server to route messages through SMTP.

Before you begin

Configure a user account on the server so that emails can be sent from an address similar to `build@mycompany.com`. You must have administrative permissions to configure mail servers.

1. On the left navigational panel, click **System > Mail Server** in the **Notifications** area.
2. Click **Edit**.
3. Type a name for the mail server in the **Name** field.
4. Type the host name for the mail server in the **Mail Host** field.
   The IP address or network name that can be resolved from the Build server.
5. Enter the port number for the mail server in the **Mail Port** field.
   The default value is 25.
6. Enter the sender's email address in the **Sender** field.
   This value appears in the email `From` header.
7. Enter the user name in the **User Name** field.
   This name is authenticated by the SMTP server. It is the name that is used to log on to the SMTP server.

   Note: Some email servers and firewalls consider emails with different sender and user names as suspicious. If this problem occurs, use the same name in both fields.

8. Enter the password that is associated with the user name in the **Password** field.
9. **Optional:** To use TLS, select **Use TLS if available**.
10. Click **Done**.

Configuring instant messaging

Build can use XMPP-type messages for user notifications.

Before you begin

Configure an XMPP messaging server to route messages.

About this task

You can configure notifications that send instant messages to users when certain events occur. See Creating notification schemes on page 312.

1. On the left navigational panel, click **System > XMPP IM** in the **Notifications** area.
2. Click **Edit**.
3. Type a name for the IM service in the **Name** field.
4. Type the host name for the XMPP server in the **Server Host** field.
   The IP address or a host name that can be resolved from the Build server.
5. Enter the port number for the server in the **Server Port** field.
   The default value is 5222. Separate multiple ports with commas.
6. Enter the XMPP user domain in the **Domain** field.
7. Enter the user name in the **User Name** field.
   This name is authenticated by the XMPP server.
8. Enter the password that is associated with the user name in the **Password** field.
9. Click **Done**.

What to do next

Click **Test** to test the configuration. The test results are displayed in the **Service Status** field.

Creating notification schemes

Notification schemes define who is notified, when they are notified, and how notifications are sent.

Before you begin

To manage notifications, ensure that the following conditions are met:

- You must have read and write permissions to the **System** page.
- At least one user must be created, see Security on page 316.
- To send emails, an email server must be configured, see Configuring mail servers on page 311.
- To send instant messages, an XMPP service must be configured, see Configuring instant messaging on page 312.

About this task

Notification schemes define rules that determine which user groups are notified about selected events.

Note:

1. On the left navigational panel, click **System > Notification Schemes** in the **Notifications** area.
2. Click **Create Create Notification Scheme**.
3. Using the **New Notification Scheme** page, type a name for the scheme in the **Name** field, and click **Save**.
4. Click **Add Who-When** and complete the following actions.

    a. Select the event that triggers notifications from the **Event Selector** list.
    Build provides several event selectors. For information about creating your own event selector, see Configuring event selectors on page 315.

    b. Select the recipient generator that determines who receives notifications from the **Recipient Generator** list.
    Build provides several recipient generators. For information about creating your own recipient generator, see Creating recipient generators on page 314.

    c. **Optional:** Select the recipient generator that determines who receives CC notifications from the **CC Recipient Generator** list.

    d. **Optional:** Select the recipient generator that determines who receives BCC notifications from the **BCC Recipient Generator** list.

5. Click **Save**.

Creating notification templates

Notification templates determine how users are notified about events.

About this task

Notification templates are Apache Velocity templates that create documents from build results. Templates are paired with events. Different templates are appropriate for different mediums. A template for instant messages, for example, does not have a subject section. Build provides various templates for you to use and customize.

1. On the left navigational panel, click **System > Notification Templates** in the **Notifications** area.
2. Click **Create New Template**.
3. Using the **New Email Template** page, type a name for the template in the **Name** field.
4. Enter the template in the **Template Text** area.
    Templates are written in the Velocity template language. Each template must contain a Subject and Body section.
    Example
    Here is an example template. The Build Request Failed Template is one of the templates that are provided with the product.

```
## BEGIN SECTION Subject

#set($trace = $request.JobTrace)
#set($project = $trace.Project)

Request Failed: $project.Name - $request.Workflow.Name
```

```
## END SECTION Subject

## BEGIN SECTION Body
## PROPERTY Content-Type: text/html
## PROPERTY X-Priority: 3

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

#set($trace = $request.JobTrace)
#set($project = $trace.Project)

<html>

<head>
<title>Process</title>
</head>
<body>

Request Failed: $project.Name - $request.Workflow.Name
<p>

Additional information available here:
<a href="$request.Url">$request.Url</a>

</body>
</html>

## END SECTION Body
```

5. Click **Set**.

Creating recipient generators

Recipient generators determine which users are notified about various events.

About this task

Recipient generators can generate lists where the recipients are fixed and invariable, or lists where recipients vary. The recipient generators provided with the product determine recipients that are based on user roles. For information about user roles, see Roles and permissions on page 319.

1. On the left navigational panel, click **System > Recipient Generators** in the **Notifications** area.
2. Click **Create Recipient Generator**.
3. Using the **New Recipient Generator** page, select the type from the **Type** list, and click **Set**.
   Fixed types expect you to directly enter recipient information, such as email and IM addresses. Group-Based types use security groups to generate recipients. Scripted types use JavaScript scripts to generate recipients.

   Note: To give users the ability to unsubscribe from notification subscriptions, use group-based or scripted types. If a process has a fixed type, users cannot unsubscribe from notification subscriptions.

4. Type a name for the generator in the **Name** field, and click **Save**.

5. Define the generator by completing these actions:

   Choose from:

   - If you selected the **Fixed** type, enter the user's email address in the **Email** field, or XMPP ID in the **XMPP** field, then click **Add**.
   - If you selected the **Group-Based** type, select the groups that you want notified from **Groups** area.
   - If you selected the **Scripted** type, enter the script in the **Script** box. The script can be any JavaScript script that resolves to a user or group. This example script selects all users: `users.getAllUsers()`.

6. Click **Done**.

What to do next

Add the recipient generator to notification schemes, see Creating notification schemes on page 312.

Configuring event selectors

Event selectors are used to select events that trigger user notification.

About this task

Build provides a selection of events that can trigger notification. The provided scripts examine events and return a value of true if a notification is triggered, or false otherwise.

1. On the left navigational panel, click **System > Event Selectors** in the **Notifications** area.
2. Click **Create Event Selector**.
3. Using the **New Event Selector** page, type a name for the template in the **Name** field.
4. Select the event that you want to use as a trigger from the **Event Type** list.

   The following list describes the event types that are provided with the product:

   - **Agent Status Changed** triggers when an agent changes status, such as Online or Offline.
   - **Build Deleted** triggers when a workflow is deleted.
   - **Build Status Applied** triggers when a build status is added or changed.
   - **Build Request Failed or Build Not Needed** triggers when a workflow fails or does not start because of lack of source changes.
   - **Build Not Needed** triggers when a workflow does not start because of lack of source changes.
   - **Build Request Failed** triggers when a workflow does not occur due to an error in the build request.
   - **Workflow End** triggers notifications when the workflow ends regardless of status.

5. Enter the script in the **Scripted** box.

   The script can be any JavaScript script that resolves to a true or false value.

   Example

   The Workflow Status Change script triggers when a workflow ends successfully after the previous invocation failed, or fails after the previous invocation was successful. This script is one of the scripts that are provided with the product.

```
var result = false;
var prevBuildLife = workflow.getBuildLife().getPrevious()
if (prevBuildLife == null) {
    result = true
}
else {
```

```
    var prevWorkflowCase = prevBuildLife.getWorkflow()
    if (prevWorkflowCase == null || !prevWorkflowCase.isComplete() ||
 prevWorkflowCase.statusIsSuccess() != workflow.statusIsSuccess()) {
        result = true
    }
}

result
```

6. Click **Save**.

What to do next

Add the event selector to notification schemes, see Creating notification schemes on page 312.

Security

Build provides a flexible team-based, and role-based security model that maps to your organizational structure.

From a high level, the security system consists of an authentication realm, authorization realm, roles, and teams. The authentication realm verifies the identity of the user or system that is trying to log on to Build. The authorization realm manages user groups. Roles manage permissions. Teams join users with roles and secure product areas.

Overview of permissions and security types

A good starting point for learning about the team-based security system is to understand permissions. Build uses the term *permission* in a way that is familiar to most readers: each permission controls access to a product area or product function. Most permissions refer to typical activities such as creating, reading, updating, and deleting. Permissions define *what* can be done, not *who* can do it

A product area that can have defined permissions is referred to as a *security type* or *security resource*. Each security type has a set of permissions that affects how users interact with it. The project security type, as the name implies, has a set of permissions that affects a user's ability to interact with projects; the permissions that are available for the agent pool security type affect agent pools, and so forth.

Most security types can have additional types defined for them. Each eligible security type comes with a "standard" type that is defined for it, such as "Standard Process". Each additional type within an area shares the same set of permissions. For example, each agent pool type has the same set of permissions: edit and view. See Security types on page 318.

Roles

A *role* is a set of granted permissions. Defining a role consists of granting permissions for all security types that the role affects. The number of roles and their functions are completely up to you. As delivered, Build provides one role, the administrator role, which has all permissions granted for all security types.

Importantly, a role by itself does not impart its granted permissions to any actual user. Like permissions, roles define what can be done, not who can do them. Roles and their associated permissions are applied to users by teams.

Teams

A *team* is a construct that associates users or groups with roles. When a user is added to a team, that person is assigned to a role; users cannot be added to a team without a role assignment. Role members are automatically granted all permissions that are defined for the role. Groups can also be added to roles, in which case all group members are automatically granted the permissions that are defined for the role.

When a team is created, all users and groups are available for assignment, and all roles are available for use. Users can be assigned to more than one role. If a user is assigned to one role that grants a particular permission and another role that does not grant that permission, the user is considered to have the permission.

Teams secure resources. To secure a workflow, for example, a team is associated with it. After being secured, only team members who are assigned a role with an appropriate permission can affect the workflow.

Securing resources

The Build security system enables fine-grained customizations. The way all elements come together might not be clear. A further discussion about permissions might provide more clarity.

Two types of permissions are available: types with specific scope; and types with a more general scope. The Job security type has three permissions: create, edit, and view. Except for create, these permissions affect only a specific resource; they are scoped to a specific job. If a team with the edit permission is attached to Job A, for example, then only team members with that permission can edit Job A. Team members with other roles that do not grant the edit permission cannot edit the job. Members of teams that are not attached to Job A cannot edit it regardless of whether they have the edit permission for other jobs.

The create permission is different from other permission types. The create type is not scoped to a specific resource but is more broadly scoped. Team members with a create-type permission can create resources of the related type without first being attached to a specific resource. To continue the job example, users with the create job permission can create jobs even if their team is not attached to a specific job.

Guidelines for setting up security

Although there is no single best way to set up security, the following steps are sufficient in most cases.

1. Create an authorization realm. Authorization realms are used by authentication realms to associate users with groups and to determine user access. Build includes an internal database for storing security information as well as integration with the Lightweight Directory Access Protocol (LDAP). LDAP is a widely-used protocol for accessing distributed directory information over IP networks. See Authorization realms.
2. Create an authentication realm. The authentication realm is used to determine a user's identity within an authorization realm. See Authentication realms.
3. Create roles and define permissions for them. For most uses, the default security types are adequate. See Roles and permissions on page 319.
4. Create and import users and groups.
5. Create teams and assign users and groups to them. See Security teams on page 320.

Security types

Security types have permissions that define the actions that users can do.

A *security type* or security resource, is a product area that can have permissions defined for it. Each security type has a set of permissions that affects how users interact with it. Most security types can have additional defined types, which are sometimes called *subtypes*. By default, each type has a defined subtype, which is called the Standard type. The following table lists security types and permissions.

Table 8. Security types

Security type	Permissions
Agent pool	• **edit:** Modify and delete agent pools • **view**: Access agent pools in the UI but cannot modify them
CodeStation	• **create**: Create CodeStation projects • **edit**: Modify and delete CodeStation projects • **view**: Access CodeStation projects in the UI but cannot modify them • **download**: Download artifacts from CodeStation
Job	• **create**: Create jobs • **edit**: Modify and delete jobs • **view**: Access jobs in the UI but cannot modify them
Project	• **create**: Create projects • **edit**: Modify and delete projects • **view**: Access projects in the UI but cannot modify them • **artifact download**: Download artifacts from CodeStation or other repositories
Project process	• **add manual stamp**: Apply manual time stamps • **add manual status**: Apply manual statuses • **delete history**: Delete project history • **abort**: Cancel running project processes • **artifact download**: Download project artifacts from CodeStation or other repositories • **prioritize**: Determine run priority when a project has two or more processes • **restart**: Restart project processes • **run**: Run project processes

Table 8. Security types (continued)

Security type	Permissions
Template	• **create**: Create templates • **edit**: Modify and delete templates • **view**: Access templates in the UI but cannot modify them
Report	• **edit**: Modify and delete reports • **run**: Run reports
Repositories	• **create**: Create repositories • **edit**: Modify and delete repositories • **view**: Access repositories in the UI but cannot modify them
Security	• **Agent management**: Create, modify, and delete agents • **Agent view**: Access the Agents page in the UI • **Audit administration**: Run audit reports for resources and users • **Report create**: Create reports • **Scripts administration**: Add, modify, and delete scripts in the Script Library • **Security administration**: Create, modify, and delete authorization and authentication realms • **System administration**: Modify system-related settings, such as license servers, properties, and server settings • **Team resource Management**: Assign security types to teams, such as projects and workflows • **Team user management**: Add and remove users to/from teams • **User Management**: Create, modify, and delete users

Creating security types

Security types can have additional defined types, which are called subtypes.

1. On the left navigational panel, click **System > Resource Types**.
2. From the list of resource types, select the one to which you want to add a subtype.
3. Click **Create New Type** to open the **Create New Type** dialog box.
4. Type a name for this type in the **Name** field.
5. Click **Save**.

Roles and permissions

Roles provide permissions to users.

A *role* is a set of permissions. Typically, the permissions in a role define a particular activity that a user might do. Users can be assigned to more than one role. The number of roles and their permissions is up to you. Build provides one role, the Administrator role, which has all available permissions.

Users are granted permissions by being assigned to roles. When assigned to a role, a user is automatically granted all permissions that are defined for the role. Typical activities include changing or running an item, such as a project process, or modifying security settings.

Note: If a role has the create permission for some security type, such as projects, the role typically grants the edit permission for that type.

Creating roles

Follow these steps to create roles and assign permissions to them.

1. On the left navigational panel, click **System > Roles**.
2. Click **Create New Role**.
3. Type a name for the role in the **Name** field, and then click **Save**.
4. Select the role to display the list of security types and permissions.
5. With the role selected, click the security type to use. The available permissions for the security type are displayed.
6. Grant permissions by selecting the associated check boxes.
7. Click **Done**.

Note: If a role has the create permission for some security type, such as projects, the role also grants the edit permission for that type.

Security teams

Through teams, users are assigned to roles and can secure product resources.

Teams provide two important functions. First, a team provides the mechanism to assign users roles. When a user is assigned a role, all permissions that are granted to the role are automatically granted to the user. Second, teams secure product resources. When a team is attached to a resource, only team members with the appropriate permissions can interact with the affected resource.

Creating teams

Follow these steps to create teams.

1. On the left navigational panel, click **Teams**.
2. Click **Create New Team**.
3. In the **Create New Team** page, type a name for the team in the **Name** field.
4. Click **Save**.

Assigning users to teams

To add users to teams, select the users and assign them to a role.

1. On the left navigational panel, click **Teams**.
2. In the team list, select the team to add users to.

   When a team is selected, all roles that were previously defined are available. All previously defined roles are displayed and users are listed below their assigned roles.
3. To add users to roles, do one of these actions:
   Choose from:
   - Click **Add** button for the target role, and then select the check boxes from the list of users.
   - Click **Add Users and Groups** button, and then drag users onto target roles.

What to do next

Assign teams to resources, such as projects.

Mapping teams to resources

In addition to assigning a team to a resource (such as a workload or job) while you create the resource, you can use the **Teams** page to map teams to resources.

Before you begin

To map a team to an object, you must be a member of the team and have the manage teams permission for that type of object.

About this task

You can assign teams to many items that you create. Teams can be assigned when you create an item in the associated **Create New item** window. Teams can also be mapped to objects with the object mapping security feature as described here. When a team is assigned to an item, such as a project, only team members with the appropriate permissions can interact with the affected item.

Note: All user-created resources are automatically assigned to the System Team. Because the System Team's Administrator role has all permissions that are granted for all Standard security types, System Team users with that role automatically have all permissions for all objects. Initially, only the admin user is assigned to the Administrator role. Neither the System Team nor the admin user can be deleted.

1. On the left navigational panel, click **Teams**.
2. Select a team from the list of teams.
3. Under **Team Object Mappings**, in the **View** list, select the object type, such as **Project**.
   Result
4. From the **Type** list, select the security type.
   Types for the current object type are available. For information about security types, see Table 8: Security types on page 318.

5. Click **Add**.
   A list of user-created objects that are not already assigned is displayed.
6. Select the check boxes for the objects to assign to the team and then click **OK**.

Results

Teams that are mapped to objects are displayed on the associated **Security** page for the object.

Viewing resources that are assigned to teams

You can view the resources that are assigned to a team.

1. Click **Teams**.
2. In the **Teams** pane, select a team from the team list.
3. Under **Team Object Mappings**, select the product area to view from the **View** list.
   Example
   For example, select **Workflow**.
   All objects for the selected product area that are mapped to the team are displayed.

Viewing teams that are assigned to resources

You can view the teams that are assigned to a resource.

1. Select the object to review, such as a project or workflow.
2. On the page for the object, click **Security**.
   Result
   The **Resource Security** window displays the roles and users that are granted permissions. For example, the **Resource Security** window for project templates lists the roles and users that can create, edit, and view the selected template.

Auditing team activity

You can review team activity for many product areas.

About this task

You can review activity for the entire team or for a single team member.

1. On the left navigational panel, click **Teams > Auditing**.
2. Select a team from the **Team** list.
3. Select a user from the **User** list.
   To see activity for all users, select **All Users**.
4. In the **Start Time** field, enter the first date to review activity for.
   The format is `yyyy-mm-dd`.
5. In the **End Time** field, enter the last date to review activity for.
6. Click **Search**.

Users and groups

Users and user groups can be created manually or imported from external systems, such as LDAP.

For information about importing users, see Creating authentication realms.

For information about importing groups, see Authorization realms.

Manually creating groups

Groups are logical containers that serve as a mechanism to grant permissions to multiple users; members automatically share a group's permissions.

About this task

You can create groups manually for internal storage type authorization realms. Groups must be imported for others types, such as LDAP.

1. On the left navigational panel, click **System > Groups**.
2. Click **Create Group**.
3. Type a name for the group in the **Group Name** field.
4. Select an authorization realm from the **Authorization Realm** list.
   Groups are only valid for the selected realm.
5. Click **Save**.

What to do next

Add users to the group. See Manually adding users to groups.

Creating users manually

You can add users manually to internal-type authentication realms.

1. On the left navigational panel, click **Teams > Users > Create User**.
2. If you have more than two authentication realms, select a realm from the **Authentication Realm** list.
3. Type a user name to be used to log into the user interface for the user in the **Name** field.
4. Type a password for the user in the **New Password** field, and then reenter the password in the **Confirm Password** field.
5. From the **Groups** area, select the groups to add the user.
6. Type the user's name in the **Actual Name** field.
7. Type the user's email address in the **Email Address** field.
8. Type the user's instant messaging ID in the **IM ID** field.
9. Select time and date format to be used for the user:
   a. Select the user's preferred date and time format from the **Date and Time Format** list.
   b. Select whether the time format is based on the 24-hour clock or 12-hour clock.
10. Select the user's time zone from the **Time Zone** list.
11. Specify the number of items that are displayed on the user's dashboard in the **Number of Dashboard Rows** field.

12. **Receive Notifications**
13. Click **Save**

Deleting users manually

Users can be permanently removed. To remove a user, you must be assigned either the **User Management** or **Security Administration** permission. Before removing a user, you should verify that the user is no longer needed or created in error. A user will not be deleted if there is any history associated with the user.

1. On the left navigational panel, click **Teams > Users**.
2. Click the delete icon ( ) to the right of the user to be deleted.
3. A confirmation message displays. Click **OK** to delete the specified user.
   An error message is displayed if history is associated with the user.

User profiles

Information about users is maintained in a *user profile* for each user. The information can be viewed and modified from the dashboard.

The user profile contains the following information:

- User name and password
- Name of the user
- Authentication realm of the user
- Groups the user is associated with
- User email address
- Date and time format preference of the user
- Time zone the user is located in
- Geographical location of the user
- Amount of information to show on the dashboard
- Notification preference for receiving information from the server

The information that is defined in the user profile can be modified by an administrator. Some of the values can also be modified by the user.

Updating your user profile and password

You can change your user profile information and password .

1. At the upper-right corner of the server window, click your user name, and then click **Edit Profile**.
   The **User Profile** displays.
2. Click **Profile**, and then click **Edit**.
3. You can modify the following fields.

User Name

The user name to be used to log into the user interface for the user in the **Name** field.

Name

The full name of the user.

Email Address

The email address of the user.

Instant Messaging ID

Time and Date Format

The time and date format used by the user and whether the time format is based on the 24-hour clock or 12-hour clock.

Locale

The geographical location of the user.

Number of Dashboard Rows

The number of items that are displayed on the user's dashboard.

Receive Notifications

Enable to receive notifications from the server.

4. Click **Save**.

Changing your password

1. At the upper-right corner of the server window, click your user name, and then click **Edit Profile**.
   The **User Profile** displays.
2. On the **General** page, click the **Update Password**.
3. Type your current password in the **Old Password** field.
4. Type a new password in the **Password** field, and then type it again in the **Confirm Password** field.
5. Click **Update** to save the new password.

Managing user profiles

The administrator can modify user profiles and reset passwords for internal authentication realm-based user accounts.

Before you begin

You must have administrator access to update another user's profile or reset a user's password.

1. On the left navigational panel, click **Teams > Users** and then select a user from the **User Name** list.
   The **Access** page displays the teams that the user belongs to, and the resources that those teams manage.
2. Click the **Profile** tab, and then click **Edit**.
3. Select the user that the profile is being updated.

4. From the **Groups** area, select the groups to add the user.
5. Type the user's name in the **Actual Name** field.
6. Type the user's email address in the **Email Address** field.
7. Select time and date format to be used for the user:
    a. Select the user's preferred date and time format from the **Date and Time Format** list.
    b. Select whether the time format is based on the 24-hour clock or 12-hour clock.
8. Select the user's time zone from the **Time Zone** list.
9. Specify the number of items that are displayed on the user's dashboard in the **Number of Dashboard Rows** field.
10. Click **Save**.

Resetting user passwords

1. From the Build dashboard, click **Teams**, click **Users**, and then select a user from the **User Name** list.
2. Click the **Profile** tab, and then click **Reset Password**.
3. Type a password for the user in the **New Password** field, and then reenter the password in the **Confirm Password** field.

Creating tokens

Tokens provide authorization for agents, users, and external systems or applications. Agents use tokens when they run process steps and communicate with the Build server and external services.

You can create tokens in addition to the tokens that the product provides.

1. On the left navigational panel, click **System > Auth Tokens > Create New Token**.
2. Type a user name in the **User** field.

    Note:

    If you are using the token to integrate with HCL DevOps Deploy (Deploy), you must have the necessary permissions. The Build user account must have permission to create component versions, set statuses on versions, and upload artifacts to component versions.

3. Specify the expiration date and time.

Tokens can be used immediately after they are created.

Auditing user activity

You can review user activity for many product areas.

About this task

You can review activity for one user or all users. You can review activity for all object types or a single type, such as workflows and projects.

1. On the left navigational panel, click **System > Auditing**.
2. Select the product area to review from the **Object Type** list.
   To see activity for all types, select **All Object Types**.
3. Select a user from the **User** list.
   To see activity for all users, select **All Users**.
4. Enter the first date to review activity in the **Start Time** field.
   The format is `yyyy-mm-dd`.
5. Enter the last date to review activity in the **End Time** field.
6. Click **Search**.

Project support

Projects represent items that can be built individually for release or distribution.

Projects often correspond to how development is structured in your source control management system. For example, projects can be composites of many other projects using a dependency management system. Projects are built from templates.

Creating artifact sets

Artifact sets are groups of artifacts that are produced by a build.

Before you begin

Typically, there is an artifact set for each type of artifact. Run-time-dependent artifacts, such as `jar` or `dll` type files, might be contained in an artifact set called lib. Source code artifacts might be contained in an artifact set called source.

1. On the left navigational panel, click **System > Artifact Sets**.
2. Using the **Artifact Sets** page, click **Create New**.
3. Using the **New Artifact Set** dialog box, type a name for the workflow in the **Name** field.
4. Optionally, enter a description of the artifact set in the **Description** area.
5. Click **Save**.

What to do next

Artifact sets can be configured when you create build process templates. See Creating build process templates on page 388.

Managing cleanups

Cleanups are scheduled based on the cleanup policy. The cleanup policy contains settings that determine when build lives, build requests, and miscellaneous jobs are cleaned up.

About this task

Build life cleanup is determined by cleanup type and retention policy. Each build life is evaluated by its status. A build without a status is evaluated with the **All Builds** status. If the **All Builds** type has the **Keep Days** property or **Keep Latest** property set

to `Forever`, no builds with this type are be cleaned up. A build life is not cleaned up if either the **Keep Days** or **Keep Latest** properties are met.

Complete the following steps to schedule a cleanup.

1. On the left navigational panel, click **System > Cleanup**.
2. Click **Edit**.
3. Select a schedule from the **Schedule** list.
   Schedules are defined on the **Schedules** page, see Creating schedules on page 332.
4. Specify a number of days to keep lockable resources in the **Days to Keep Lockable Resources** field.
   This option applies to resources such as working directories. The default value is 30 days. Enter `-1` to never clean up lockable resources.
5. Specify the number of days to keep preflight builds in the **Days to Keep Preflight Builds** field.
   The default value is 30 days. Enter `-1` to never clean up lockable resources.
6. Click **Save**.
7. **Optional:** Create additional cleanup policies.
   Click **New** to create a cleanup policy. See Adding and editing cleanup policy settings on page 328 for details on cleanup policy settings.
8. Click **Done** after you finish.

Related information

Adding and editing cleanup policy settings on page 328

Adding and editing cleanup policy settings

Use the cleanup settings to determine when build lives, build requests, and miscellaneous jobs are cleaned up.

About this task

You can define multiple cleanup policies and modify the default policy. The list of policies is displayed on the **Cleanup** window. Policies that you create can be deleted using the delete icon. The default policy can be disabled.

1. On the left navigational panel, click **System > Cleanup**.
2. Click **New** to add a policy. To modify a cleanup policy, click the policy name.
   If you are creating a new policy, complete the **Name** and policy settings are displayed.
3. To modify a build cleanup settings, click the edit icon in the **Cleanup Build** area.

   a. Specify the number of days to keep build lives in the **Keep Days** field.
      Leave the field blank to keep builds forever.

   b. Specify the number build lives to keep regardless of other cleanup settings in the **Keep Latest** field.
      The number of build lives that are kept indefinitely is equal to the number you specify.

c. Select the cleanup type from the **Cleanup Type** list.
   The cleanup type determines what happens to build lives with a certain status when a cleanup is scheduled:
   - **Delete** deletes the build life, artifacts, and logs from the server.
   - **Deactivate and Delete Logs** deletes the build artifacts and logs and sets the build to inactive status. This option prevents running processes on the build life and prevents the use to satisfy dependencies.
   - **Deactivate** deletes the build life artifacts and sets the build life to inactive status. This option prevents process runs and prevents the use of the build life to satisfy dependencies.
   - **Archive** deletes the build life artifacts and sets the build life to archived status. The build life can be unarchived and its artifacts rebuilt with an unarchive process. No secondary processes can run on this build and it cannot be used to satisfy dependencies.

d. Click **Save**.

4. To modify a build request cleanup settings, click the edit icon in the **Cleanup Build Requests** area.

   a. Specify the number of days to keep miscellaneous jobs in the **Misc Jobs** field.
      Miscellaneous jobs refer to cleanups and notifications among others. These jobs and their logs can be deleted from the server by specifying a number to keep. Leave the field blank to keep them forever.

   b. Specify the number of days to keep build requests without builds in the **Build Requests Without Build** field.
      These build requests did not produce builds. Leave the field blank to keep requests forever.

   c. Click **Save**.

5. Click the **Dependencies** tab, and then specify projects that depend on the policy.
6. Click the **Main** tab and then click **Done**.

Managing Labels

Labels are assigned to build lives. Using the **Label** page, you can create, modify, and delete labels. Deleting a label removes the label from the server.

1. On the left navigational panel, click **System**, then click **Labels** located under **Project Support**.
2. To create a label, click **Create New Label**.
   a. Type a name in the **Name** field.
   b. Optionally, type a description in the **Description** field.
   c. Click **Save**.
3. To modify an existing label, click the label name in the table.

   a. Change the existing name in the **Name** field. You can also update the text in the **Description** field.

   b. Click **Save**.
      The new label name is displayed in the table.

4. To delete a label, click the delete icon ( ) for the label to be deleted.

a. Click **OK** to confirm the deletion.
   The deleted label no longer exists on the server causing it to be removed from all associated build lives.

Creating lockable resources

Lockable resources are resources that can be reserved by other objects, such as agents, when a process runs.

About this task

Locked resources cannot be accessed by other objects after the lock is acquired. Other requests for the resource are queued until the lock is released.

1. On the left navigational panel, click **System > Lockable Resources**.
2. Click **Create New**.
3. Using the **Edit Lock** dialog box, type a name for the resource in the **Name** field.
4. Specify the maximum number of simultaneous lock holders in the **Maximum Number of Lock Holders** field.

   The default value is 1, which means that only one process can obtain a lock on the resource. If you specify 2, for example, a maximum of two different processes can concurrently use the resource. This value can be overridden on individual processes.
5. Click **Save**.

What to do next

Lockable resources must be added to processes. Resources can be configured when you create build process templates.
See Creating build process templates on page 388

Repositories

Repositories contain the source materials that are used in build processes.

Repositories are configured from source plug-ins. Initially, the File System and Maven plug-ins are available. If you want to create other types of repositories, install the corresponding source plug-ins. For information about installing plug-ins, see Installing plug-ins on page 351.

Note: File System and Maven are not derived from source plug-ins, and cannot be configured like other repositories derived from source plug-ins.

Creating file system repositories

File system repositories manage files that are stored in file systems.

1. On the left navigational panel, click **System > Repositories**.
2. Click **Create New**.
3. Using the **Create Repository** page, select a plug-in from the **Source Plug-in** list, then click **Set**.
4. Type a name for the repository in the **Name** field.

5. Enter a description the repository in the **Description** area.
6. Select a team to manage the repository from the **Teams** list.
7. Click **Save**.

What to do next

Source integrations are implemented as steps for build jobs. After you configure a repository, you can add steps from the associated source plug-in to jobs.

Creating Maven repositories

Build can act as a Maven repository for projects that are built in Build, and can serve as a caching proxy for external Maven repositories.

Before you begin

Before you start the integration, ensure that these preconditions are met:

- The build agent must be in the same location as Maven (or Maven must be downloaded with the project).
- You must have administrative permissions to configure projects and have access to the Build **System** page.
- You must be able to edit the Maven `settings.xml` file, which is usually located in the */user.home/*`.m2/` directory. By default, this file does not exist and Maven uses default values for configuration properties. If you create the file and place it in the location specified, Maven usees it instead of the default values. Maven provides an sample `settings.xml` for you to use and modify, which is found in the *maven_install_dir*`/conf` directory.
- You must be able to edit the Maven POM file, which is usually located in the Maven application's root directory.

Note: The File System and Maven source configurations are not derived from source plug-ins, and cannot be configured like other repositories derived from source plug-ins.

Modify the Maven POM file to reference the repository that is used by Build. The POM file should look similar to this example:

```
<project>
 <distributionManagement>
    <repository>
      <id>DevOps_Build</id>
      <name>DevOps Build Repo</name>
      <url>${env.DEVOPS_BUILD_WEB_URL}rest/maven2dist/${env.DEVOPS_BUILD_BUILD_LIFE_ID}</url>
    </repository>
    <snapshotRepository>
      <uniqueVersion>false</uniqueVersion>
      <id>DevOps_Build</id>
      <name>DevOps_Build Repo</name>
      <url>${env.DEVOPS_BUILD_WEB_URL}rest/maven2dist/${env.DEVOPS_BUILD_BUILD_LIFE_ID}</url>
    </snapshotRepository>
  </distributionManagement>
</project>
```

Modify the `settings.xml` file in the location where the agent that is accessing Maven is located. The file must provide a user name and password for Maven authentication, similar to the example in the following code fragment:

```
<settings>
  <servers>
    <server>
      <id>DevOps_Build</id>
      <username>username</username>
      <password>password</password>
    </server>
  </servers>
</settings>
```

About this task

With this integration, you can retrieve artifacts from CodeStation or directly from your Maven repositories. The integration provides visibility into the project dependencies you manage using Maven 2.

1. On the left navigational panel, click **System**, then click **Maven** under **Integrations**.
2. In the **New Maven Repository** dialog, type a name for the repository in the **Name** field.
3. Enter a description the repository in the **Description** area.
4. Type the URL for the Maven repository in the **URL** field.
   The format is `[HOST]:[PORT]`; for example, `http://maven.domain.com:8080`.
5. Type the Maven user name fin the **User Name** field.
6. Type the password for the account in the **Password** field.
7. **Optional:** To configure a proxy server, see Creating proxy servers on page 336.
8. To verify transferred Maven files, select **Verify Hashes**.
9. Click **Save**.

Creating schedules

You can use schedules to determine when certain events, such as builds and backups, occur.

Before you begin

To create a schedule, you must have administrative permissions.

About this task

A schedule can be used in many product areas. For example, you can create a schedule that triggers nightly builds and add it to a project trigger. Schedules are used with cleanup options, see Managing cleanups on page 327. You can create two types of schedules:

- Interval-based schedules use a fixed time interval to determine when events occur.
- Cron-based schedules can use complex criteria to define when events occur.

1. On the left navigational panel, click **System > Schedules**.
2. Click **Create Schedule** then define the schedule type:
   Choose from:
   - To create a cron-based schedule, select **cron expression**.

   Choose from:
   - To create an interval-based schedule, select **interval schedule**.

3. **Optional:** If you selected **cron expression**, complete the following steps.

    a. Type a name for the schedule in the **Name** field

    b. Enter the cron expression in the **Cron Expression** field.

    A cron expression is a string of six mandatory properties and one optional properties. Properties must be defined in the order they are presented in the table. Separate properties with white spaces. Use the following table to complete the expression:

    Table 9. Cron expression properties

Name	Expected values
Seconds	0-59
Minutes	0-59
Hours	0-23
Day of Month	1-31
Month	1-12 or JAN-DEC
Day of Week	1-7 or SUN-SAT
Year (optional)	empty, 1970-2099

    The following special characters can be used in expression properties:

    Table 10. Special characters

Character	Description
*	All values. For example, "*" in the minute field means every minute.
?	Used in the day-of-month and day-of-week fields and means no specific value. Use this character when you need to specify something in one field but not another.
-	Specifies a range of values. For example `10-12` in the hour field means the hours 10, 11 and 12.
,	Specifies additional values. For example `MON,WED,FRI` in the day-of-week field means the days Monday, Wednesday, and Friday.
/	Specifies increments. For example `0/15` in the seconds field means the seconds 0, 15, 30, and 45.
L	Used in the day-of-month and day-of-week fields. Short-hand for last, but it has a different meaning in the 2 fields. For example, the value `L` in the day-of-month field means the last day of the month. Used in the day-of-week field by itself, it means 7 or SAT. Used in the day-of-week field after another value, it

Character	Description
	means the last x day of the month. For example, `6L` means the last Friday of the month. When you use this character, do not specify lists, or ranges of values.
W	Used in the day-of-month field. Specifies the weekday (Monday-Friday) nearest the day. For example, `15W` means the nearest weekday to the 15th of the month. If the 15th is a Saturday, the trigger is on Friday the 14th. If the 15th is a Sunday, the trigger is on Monday the 16th. If you specify `1W` and the first is a Saturday, the trigger is on Monday the third because it does not cross the month boundary. Cannot be used for a range or list of days. `L` and `W` can be combined. For example, `LW` means the last weekday of the month.
#	Used in the day-of-week field. Specifies the nth day of the month. For example, `6#3` means the third Friday of the month (day 6 = Friday and `#3` = the third one in the month). If you specify `#5` and there is not 5 of the specified days-of-week in the month, the trigger does not occur that month.
C	Used in the day-of-month and day-of-week fields. Short-hand for calendar, which means values are calculated against the associated calendar. If no calendar is associated, it represents an all-inclusive calendar. For example, `5C` in the day-of-month field means that the first day included by the calendar on or after the fifth.

Example

`0 0 * ? * MON-FRI` means run every hour every Monday, Tuesday, Wednesday, Thursday, and Friday.

c. Click **Save**.

Result

The builds that match your search criteria are displayed.

4. **Optional:** If you selected **interval schedule**, complete the following steps.
   a. Type a name for the schedule in the **Name** field
   b. Enter the number of minutes between event triggers in the **Build Interval (minutes)** field.
   c. Enter the time when the schedule starts in the **Interval Start Tim (hh:mm)** field.
   d. Click **Save**.

Creating statuses

Statuses represent build conditions, such as success or failure.

About this task

Statuses can be used for reporting, such as finding all source changes since a status was applied. They can also be used in dependency relationships.

1. On the left navigational panel, click **System > Statuses**.
2. Click **Create New**.
3. Using the **New Status** dialog box, type a name for the status in the **Name** field.
4. Enter a description the status in the **Description** area.
5. Select a color for the status from the **Color** list.
6. Enter the number of days the status is kept in the **Keep Days** field.
   Leave the field blank to keep the status indefinitely.
7. To prevent a number of builds from being cleaned up, enter the number in the **Keep Latest** field.
   The value prevents the number of builds with this status from being cleaned up. Leave the field blank to keep all builds with this status from being cleaned up.
8. Select a clean-up type from the **Cleanup Type** list.
   For information about clean-up types, see Managing cleanups on page 327
9. Click **Save**.

Server management

Use server settings to manage your server other system-related settings.

These settings are used to configure the following features:

Creating announcements

Announcements are messages that are displayed on the Build dashboard for all users.

About this task

Announcements can be created by an Build administrator. Users can hide any announcement by clicking the **Hide** link within the announcement.

1. On the left navigational panel, click **System > Announcements**.
2. Using the **Announcements** page, click **New**.
3. Type or paste the body of the message in the **Message** area.
   The message can be up to 4,000 characters.
4. Select a priority level from the **Priorty** list.
   Priority levels affect the color of the background color of the message: **Low**=Blue, **Normal**=Green, **High**=Red
5. Click **Save**.

Creating system properties

System properties can set system-wide values for processes and projects.

About this task

The syntax for property references is `${p:property_name}`.

1. On the left navigational panel, click **System > Properties**.
2. Using the **Properties** page, click **Add Property**.
3. Type a name for the workflow in the **Property Name** field.

4. Type a value for the property in the **Value** field.

   Empty-value properties allow a template to be reused across a large number of projects, even if different projects require different values. Empty-value properties can also be used to create standardized project templates.

5. **Optional:** To secure the value, select **Secure** then type the value.
   Because secured values are obfuscated, you must also type the value in the **Confirm** field.
6. Type a description for the property in the **Description** field.
7. Click **Save**.

Creating proxy servers

Use a proxy server if you need to access servers or repositories, such as Maven, that are on external networks.

Before you begin

You must have administrative permissions to configure proxies.

1. On the left navigational panel, click **System**, then click **Proxy Settings** in the **Server** area, then click **Create Proxy**.
2. Using the **New Proxy** page, in the **Proxy Name** field, type a name.
3. Enter a description the proxy server in the **Description** area.
4. Type the host name for in the **Host** field.
5. Type the port number that Build uses to communicate with the proxy server in the **Port**.
6. Type the user name fin the **User Name** field.

   The name Build uses to connect to the proxy server.

7. Type the password for the account in the **Password** field.
8. Click **Save**.

Server settings

Use server settings to configure server-related features, such as displaying graphs on users' dashboards.

Server settings are used to configure the following features:

Configuring network settings

The properties on **Network** tab define the server's external URLs, and configure the Rational Common Licensing Server.

Before you begin
These values for the External URL and Agent External URL properties are initially defined when you install the product. The value for the Rational Common Licensing Server property can be set during installation or at any other time.

1. On the left navigational panel, click **System > Server Settings**.
2. Using the **Network** tab, click **Edit**.
3. Type the URL used by the server in the **External URL** field.
   This is the URL users use to connect to Build. Include `http(s)://` and any ports.

Example

For example, `http(s)://myserver:8080`.

4. Type the URL used by agents to access the web application in the **Agent External URL** field.
   Agents use this URL to connect to the web service to upload logs and reports. Include `http(s)://` and any ports.
   Example
   For example, `http(s)://myserver:8080`.
5. Type the location of the license server that manages your licenses in the **Rational Common Licensing Server** field.
   You can specify the port and host name or IP address for the license server, such as `27000@RCLServer.example.com`.
   To avoid problems when a license server is not available, you can specify multiple license servers. In this case, separate each address with colons on Linux™ and UNIX™ or semicolons on Windows™, as in the following example: `27000@RCLServer.example.com;27000@backupRCLServer.example.com`.
6. Click **Done**.

What to do next

The **External URL** and **Agent External URL** properties are stored in the `installed.properties` file in the `/server_install/conf/server/` directory.

Setting system-level security

Use system-level security settings to determine access to the Build server

1. On the left navigational panel, click **System > Server Settings**.
2. Click **Security**, then click **Edit**.
3. Using the following table, enable options by selecting the associated check box.

Table 11. System-level security settings

Option	Description
Audit User Transactions	When selected, the server stores security and configuration changes. The log can be accessed on the **Audit** page. The default value is unselected. See, Auditing user activity on page 326.
Allow Multiple Sessions per User	Enables users to concurrently log on from multiple computers and browsers. If cleared, the current session automatically expires if the user logs on from a second computer. The default value is selected.
Allow Use of Login Cookie	Allows users to log on with a cookie based on prior log-ins with the same computer and browser. The default value is selected.
Allow Auto-Complete Feature in Web Browsers	Allows web browsers to use auto-complete features. The default value is unselected.

Option	Description
Allow Anonymous Guest Access	Allows anonymous logon as guest user. The default value is unselected.
Show Error Trace in UI	Displays server errors in the UI. The default value is selected.
Unique Names Across All Authentication Realms	Forces all user names to be unique regardless of the authentication realm used. The default value is unselected.

4. **Optional:** Enter IP addresses and host names that can serve as repository triggers for builds in the **Allowed Addresses for Repository Triggers** area.
   Place each item on a separate line. Each item must be a regular expression. In IP addresses or host names, escape periods with the backslash character '\'. For example, `127\.0\.0\.1`. Initially, no addresses are listed.
5. Click **Save**.

Configuring miscellaneous system settings

Use miscellaneous system-level settings to manage the repository cache, dependencies, and server logging.

1. On the left navigational panel, click **Server Settings > Misc**.
2. Click **Edit** then enter the number of minutes between repository event triggers in the **Repository Trigger Cache Time** field.
   The number should equal the longest quiet period plus the longest time that is required to obtain a revision log. If the number is too small, changes might not be accurately detected when building dependent projects. The time that it takes to get the revision logs can vary significantly, depending on server load and the complexity of the dependency relationships. The default value is 30 minutes.
3. Select **Build if No Prior Successful Builds Found** if you want to build requests to produce builds even if there are no previous successful builds for the workflow.
4. Select an algorithm from the **Digest Algorithm** list.
   This option is the algorithm that is used to calculate digests when uploading artifacts to CodeStation project builds. SHA-256 is used by default. The selected algorithm is used in addition to it.
5. Select the option that is used to determine dependency conflicts from the **Dependency Conflicts** options:
   Choose from:
   - **DETECT ALL** detect all dependency conflicts even if the dependencies are not resolved.
   - **DETECT RESOLVED** only detect dependency conflicts that are resolved.
6. Select **Cascade Properties to Dependencies** if you want properties to cascade from a request to a created dependency request.
   This option applies to requests from pull and push dependencies. By default, Build does not cascade properties to requests made due to dependency relationships.
7. Select the default type from the **Default Dependency Type** list.
   This option determines the default dependency relationship for new dependencies. Users can override this setting.

8. Select the file trigger type from the **File Dependency Trigger Type** options:
   This option determines the type of dependency trigger when resolving build artifacts by using a file that defines the dependencies. If a process uses trigger-only dependencies, dependencies on the resolved projects are created with this trigger type.
9. Select **Merge Manual Builds** list to merge related manual builds.
   Requests that depend on existing requests wait for them to complete, and other requests for waiting requests are merged.
10. Select a level from the **Logging Level** options.
    The **System Properties** page lists the agent's system properties. System properties cannot be edited.
    Choose from:
    - **Info** provides basic information about server activity.
    - **Warn** logs all server warnings. The default option.
    - **Error** logs all server errors.

Placing an administrative lock on the server

Administrators can place an administrative lock on the server.

About this task

An administrative lock blocks new build requests. Blocked requests are saved and start when the lock is released. Processes that are processing when the server is locked continue normal processing. When the server is unlocked, the server status is `Normal`; when the server is locked, the status is `Administrative Lock`. Administrative locks are typically used in preparation for server maintenance.

1. On the left navigational panel, click **System > Server Settings**.
2. **Maintenance**
3. Click
4. Change the lock status with one of the following actions.
   Choose from:
   - To lock the server, click **Lock Server**.
   - To unlock the server, click **Release Lock**.

   Releasing the lock starts a recovery process that begins 1 minute after the lock is released.
5. Click **Done**.

Viewing the server log

You can view and download the server log file.

About this task

The **Log** page displays the content of the `server.out` file. The default location for this file is here: `/server_install/var/log/`.

1. On the left navigational panel, click **System > Server Settings**.
2. Click **Log**.
3. To download the log file, click **Download Log** and specify a location for the file.

Viewing the server error log

You can view and download the server error log file.

About this task

The **Log** page displays the content of the `server.err` file. The default location for this file is here: `/server_install/var/log/`.

1. On the left navigational panel, click **System > Server Settings**.
2. Click **Error**.
3. To download the log file, click **Download Log** and specify a location for the file.

Displaying graphs on the dashboard

You can display graphs on the Build user interface.

About this task

By default, graphs appear on the main dashboard and the project and process dashboards. Administrators can disable any or all of them.

1. On the left navigational panel, click **System > Server Settings**.
2. Click **Views**, then click **Edit**.
3. To disable or enable a dashboard graph, select the associated **Yes** or **No** option.
   An enabled graph displays total source activity, build activity, and average test success.
4. Click **Done**.

Running system diagnostics

You can use the diagnostics options to troubleshoot and debug various features. You can view information about active request contexts, thread CPU use, and system properties among others.

1. On the left navigational panel, click **System > Server Settings > Diagnostics**.
2. Select the options that you want to review.
   Multiple can be selected.
3. Click **View**.
   Result
   Information for the selected options is displayed.

Adding a cold standby server

Adding a *standby server* provides a backup server for disaster recovery in case the main server for Build is lost. A cold standby server is not turned on until needed. The environment must be configured to route network traffic to the standby server after a failure of the primary server.

Before you begin

- Make sure that the server is connected to a license server with available licenses.
- Ensure that you have the required security rules to get from your various endpoints to both the primary server and to the standby server.

About this task

At any given time, only one of the servers is active and connected to agents, the database, and network file systems. If the primary server encounters any issues and becomes nonoperational, then the standby server can be brought online and incoming network traffic can be directed to it instead of the primary server.

Both the primary and standby server must use the same database. The following directories and files must be moved to a network storage location that is accessible by both servers:

```
server/var
server/plugin
server/patches
server/opt/tomcat/webapps
server/opt/tomcat/work
server/conf/server.keystore
server/conf/server/base.xml
server/conf/server/ehcache.xml
server/conf/server/java.security
server/conf/server/hibernate.properties
server/conf/server/log4j.properties
server/conf/server/spy.properties
server/conf/server/rsa_api.properties
server/conf/server/upgrade.properties
```

All files and directories in the `server/opt/tomcat/work` directory should be deleted whenever a new patch is applied to the server.

1. Install the standby server in a separate location and connect it to the same database as the primary server.

   Specify the database that the production server is using when prompted for database information.

   Note: Do not start the server.

2. Set up the environment to stop the primary server, start the standby server, and direct traffic to it in the case of a failure with the primary server.
   If a load balancer is used to switch to the standby sever, set the **install.server.external.web.url** property for both server instances to the URL of the load balancer.

Agents

Agents are light-weight processes that are installed on hosts and communicate with the Build server.

Agents run commands on the host, move files to and from it, and work with other tools installed on the host. Agents are responsible for running builds and driving third-party tools. After an agent is installed, it opens a socket connection to the server for server-agent communication. Agents can also use the Build web services. After agent communication is established, agent information is displayed in the web-based UI, where it can be configured and modified.

Configuring agents

After an agent is installed, it must be configured before it can be used.

Before you begin

Install at least one agent and ensure that it is online. Once configured, access to the agent can be restricted using the security features, see Security on page 316.

1. On the left navigational panel, click **Agent > Available**.
   Agents with the status of **Available** are connected to the Build server but not yet configured.
2. Select the agents that you want to configure then use the **Configure** action .
   Result
   If configuration is successful, the agents are no longer displayed in the **Available** list.
3. Click **Configured**.
   Result
   The configured agents are displayed in the **Configured** list with the status of **Online**.

What to do next

After an agent is configured, you can modify its settings.

Modifying agent settings

After an agent is installed, you can modify its settings.

Before you begin

You must have administrative privileges to modify agent settings

About this task

Most settings are configured during installation but you can edit them any time after installation. From the **Agents List** window you can manage installed agents. The table filters at the beginning of the agent table allows you to locate a specific agent or narrow the list displayed. You can filter by agent name, host name, status, time frame the agent was used, and available, configured or ignored agents.

1. Click **Agents** and the **Agent List** window displays.
2. Click the name of the agent to view and modify agent properties.

3. On the **Main** tab, settings that define the agent and connection are displayed. You can modify the following settings. Click **Save** when finished.

   Name

   The current name of the agent was given during the installation process. The agent is automatically updated if the name is changed.

   Configured

   Enable to indicate that the agent is configured.

   Throughput Metric

   The throughput metric is an arbitrary integer that indicates the ability of an agent relative to other agents to handle extra work. An agent with a metric of 10, for example, usually gets more work before an agent with a metric of 1. A weight is applied to the agent by using a formula based on throughput and the number of active jobs. The formula is `Weight = Throughput / (Number of Active Jobs + 1)`.

   Maximum Number of Jobs

   The maximum number of concurrent jobs the agent runs. If an agent reaches the maximum number of jobs, extra requests are queued until a job finishes.

   Agents Pools

   The agent pools to which the agent is assigned.

4. Click the **Properties** tab, to view, edit, remove, and add properties to the agent.
   Agent properties are used to configure builds or other process to use values that depend on the agent that is doing the work. For example, an agent property can use an agent's local settings during a build. Agent properties are resolved at run time on the agent. Secure property values are obfuscated in logs.
   Choose from:
   - To remove a property, click the delete icon ( ) under the **Actions** column to the right of the desired property name.
   - To edit the property, click the edit icon (or). You can modify the property name and value in the **Edit Property** window. Click **Save** when finished.
   - Click **Add Property**. The **New Property** windows displays. Click **Save** when finished.

   Property Name

   Type a name for the property.

   Secure

   Click the check box to make the property secure.

   Property Value

   Type a value for the property. If you create a secure property, the typed value is obfuscated and must be retyped in the **Confirm Value** field

5. To view, edit, and create an agent environment variable, click the **Environment** tab.

Agent environment variables are set during agent installation and stored on the server.

Choose from:

- To remove an environment variable, click the delete icon ( ) under the **Actions** column to the right of the desired environment. Secure environments variables cannot be deleted.
- To edit an environment variable, click the edit icon ( ). You can modify the environments value in the **Edit Environments Variables** window.
- Click **Add Environment Variable** and the **Add Environment Variable** windows displays.

Variable Name

Type a name for the variable.

Secure

Click the check box to make the variable secure. If you create a secure variable, the typed value is obfuscated and must be retyped in the **Confirm Value** field.

Variable Value

Type a value for the property.

6. To view locked properties, click **Locked Properties**.
   The **Locked Properties** page lists the agent's locked properties. Locked properties cannot be edited.
7. To view system properties, click **System Properties**.
   The **System Properties** page lists the agent's system properties. System properties cannot be edited.
8. To set the cache location for the CodeStation repository used by the agent, click the **CodeStation** tab.
   Type the location of the cache in the **Cache Location** field. The default value is *agent_installation_directory*`/.codestation/cache`. Click **Save**.
9. Click **Done** when finished.

Related information

Configuring agents on page 342

Creating agent pools

Agent pools represent collections of agents. An agent pool can consist of one or more agents.

Before you begin

Install and configure at least one agent.

About this task

You manage agent pools on the **Agent Pool** window. A list of defined agent pools is displayed and clicking on the agent pool name details are displayed.

There are two types of agent pools: Fixed and Dynamic. You manually add agents to fixed pools. Dynamic agent pools automatically determine which agents are members of the pool.

1. On the left navigational panel, click **Agents > Agent Pools**.
2. To create a pool, click **Create Fixed Agent Pool** or **Create Dynamic Agent Pool**.
    a. Type a name for the pool in **Name** field.
    b. Enter a description in the **Description** field.
    c. Select a team to manage the pool from the **Team** list.
    d. Click **Save**.
3. To add or remove agents from a fixed agent pool, click the agent pool **Name**.
    a. Click **Add/Remove Agents**.
    b. Using the **Add/Remove Agents** window, select the agents that you want to add or remove.
    c. To add agents to the pool, select them from the **Potential Agents** list, then click the add icon .

    To remove an agent from the pool, select the agent and the remove icon ( ).
    Result
    The agents are added to the team.
    d. Click **Save**.
4. **Optional:** To review security assignments for an agent pool, select a pool then click **Security**.
   The **Resource Security** window displays the teams and users that can view and edit the pool.

Managing agent pools

Agent pools represent collections of agents. When an agent pool is assigned to a project, any pool member can do the work.

Before you begin
Install and configure at least one agent.

About this task

Agent pools help you organize and manage agents. A pool can consist of one or more agents. There is no limit on the number of agents that can be assigned to a pool. You manage agent pools on the **Agent Pool** window. In the window is a table of defined agent pools.

From the table you can view agent pool details or remove an agent pool. To remove an agent click the delete icon ( ) in the **Actions** column next to the agent pool name. To view or modify an agent pool, following the procedure below.

1. On the left navigational panel, click **Agent > Agent Pools**.
2. Click the name of the agent to view or modify.
   On the **Main** tab is a list of agents in the pool and a list of projects using the agent pool. Agent pool information is also displayed on the build life details page. From the build life you can see which agent was used by a process.
3. To modify the name of the agent or description, click **Edit**. When finished click **Save**.
4. If the agent is a fixed type pool, you can add and remove agents. Click **Add/Remove Agents**.

a. Using the **Add/Remove Agents** window, select the agents that you want to add or remove.

b. To add agents to the pool, select them from the **Potential Agents** list, then click the add icon .

c. To remove agents from the pool, select them from the **Agents in *agent_name*** list, then click the remove icon ( ).
The agents are moved to the potential agent list.

d. Click **Save**.

5. To review security assignments for an agent pool, select a pool then click **Security**.
The **Resource Security** window displays the teams and users that can view and edit the pool.

Modifying agent settings

After an agent is installed, you can modify its settings.

Before you begin

You must have administrative privileges to modify agent settings

About this task

Most settings are configured during installation but you can edit them any time after installation. From the **Agents List** window you can manage installed agents. The table filters at the beginning of the agent table allows you to locate a specific agent or narrow the list displayed. You can filter by agent name, host name, status, time frame the agent was used, and available, configured or ignored agents.

1. Click **Agents** and the **Agent List** window displays.
2. Click the name of the agent to view and modify agent properties.
3. On the **Main** tab, settings that define the agent and connection are displayed. You can modify the following settings. Click **Save** when finished.

Name

The current name of the agent was given during the installation process. The agent is automatically updated if the name is changed.

Configured

Enable to indicate that the agent is configured.

Throughput Metric

The throughput metric is an arbitrary integer that indicates the ability of an agent relative to other agents to handle extra work. An agent with a metric of 10, for example, usually gets more work before an agent with a metric of 1. A weight is applied to the agent by using a formula based on throughput and the number of active jobs. The formula is `Weight = Throughput / (Number of Active Jobs + 1)`.

Maximum Number of Jobs

The maximum number of concurrent jobs the agent runs. If an agent reaches the maximum number of jobs, extra requests are queued until a job finishes.

Agents Pools

The agent pools to which the agent is assigned.

4. Click the **Properties** tab, to view, edit, remove, and add properties to the agent.
   Agent properties are used to configure builds or other process to use values that depend on the agent that is doing the work. For example, an agent property can use an agent's local settings during a build. Agent properties are resolved at run time on the agent. Secure property values are obfuscated in logs.
   Choose from:
   - To remove a property, click the delete icon ( ) under the **Actions** column to the right of the desired property name.
   - To edit the property, click the edit icon (or). You can modify the property name and value in the **Edit Property** window. Click **Save** when finished.
   - Click **Add Property**. The **New Property** windows displays. Click **Save** when finished.

   Property Name

   Type a name for the property.

   Secure

   Click the check box to make the property secure.

   Property Value

   Type a value for the property. If you create a secure property, the typed value is obfuscated and must be retyped in the **Confirm Value** field

5. To view, edit, and create an agent environment variable, click the **Environment** tab.
   Agent environment variables are set during agent installation and stored on the server.
   Choose from:
   - To remove an environment variable, click the delete icon ( ) under the **Actions** column to the right of the desired environment. Secure environments variables cannot be deleted.
   - To edit an environment variable, click the edit icon ( ). You can modify the environments value in the **Edit Environments Variables** window.
   - Click **Add Environment Variable** and the **Add Environment Variable** windows displays.

   Variable Name

   Type a name for the variable.

   Secure

   Click the check box to make the variable secure. If you create a secure variable, the typed value is obfuscated and must be retyped in the **Confirm Value** field.

Variable Value

Type a value for the property.

6. To view locked properties, click **Locked Properties**.
   The **Locked Properties** page lists the agent's locked properties. Locked properties cannot be edited.
7. To view system properties, click **System Properties**.
   The **System Properties** page lists the agent's system properties. System properties cannot be edited.
8. To set the cache location for the CodeStation repository used by the agent, click the **CodeStation** tab.
   Type the location of the cache in the **Cache Location** field. The default value is *agent_installation_directory*`/.codestation/cache`. Click **Save**.
9. Click **Done** when finished.

Related information

Configuring agents on page 342

Restarting agent brokers

The JMS broker manages server-agent communication.

About this task

Restart brokers to resolve communication errors between agents and the server. Users with administrative permissions can restart agent brokers.

1. On the left navigational panel, click **Agent > Broker**.
2. Click **Restart Broker**.
   Result
   All existing connections between agents and the server are dropped when the broker restarts. Agents reconnect automatically after the broker starts.

Managing untrusted agent certificates

If you are using mutual authentication, the server must have a certificate in its keyStore for each agent.

About this task

Agents that do not have certificates in the server's keystore are listed on the **Agent Certificate Management** page with a status of **Untrusted**. You can override this setting.

1. On the left navigational panel, click **Agent > Agent Certificate Management**.
2. Select the agents with untrusted certificates, then click **Trust**.
   Result
   Information about the agent is displayed.
3. Click **Done**.

Viewing agent workspace data

Use the view workspace feature to view the contents of an agent's workspace related to a job for a build. The workspace viewer displays real-time content of an agent directory structure and files. You can expand directory structures and view individual file contents. There is no need to use another tool such as Remote Connect to view files on the agent.

About this task

The agent workspace viewer requires a plug-in that is included with the product. If you remove this plug-in the function becomes unavailable.

You can use the view agent workspace to troubleshoot issues and to verify that expected files were created. Log files that the agent creates might contain data that is not passed to the Build logs. This information provides additional details that are related to a failure and the build process. Output from the intermediate working files that the build job uses temporarily can be viewed on the agent. You can verify whether the files were created or corrupted while you debug issues.

It is important to understand that the viewer shows real-time data. If the workspace was cleaned as part of the build process or reused, the data that is viewed is not the job run data. If the directory is empty when you request to view it, it is empty or the agent is not online.

Access to the view workspace feature is available on several user interface pages that contain job information that is related to a build, such as the main page of a build or **Job Execution** page for a specific job.

To view agent workspace data:

1. Click the **View Workspace** icon to open the Workspace Viewer.

   The **Properties** tables contains the directory structure displayed in the Workspace Viewer window.
2. To view a specific file, navigate to the file and click **Click here to view file contents** in the right pane.
   Use the **View As** field to display the text in ASCII or EBCDIC character encoding. Binary file types are not readable.
3. To download the file, click **Download**.

Creating scripts to automate administrative tasks

The Build Client facilitates writing scripts that interact with various REST endpoints in Build to automate tasks such as creating templates, assembling projects, running builds, and various other administrative tasks.

Use the **Tools** menu to downloaded the `devops-build-client.zip` file which contains the following files:

- `devops-build-client.jar`
- API documentation
- Sample Groovy script

See Accessing development tools on page 350 for information on downloading the Build Client.

Accessing development tools

Use the **Tools** window to access the Build development tools.

About this task

There are several Build development tools that you can include in your environment.

Agent Installer

Installs build agents.

Build Client

Write scripts to create objects within the system, such as projects, project templates, and other administrative task.

CodeStation Client

Download project dependencies and build artifacts from a CodeStation repository.

Preflight Client

Initiate pre-flight builds in Build to test code before committing changes to the source code repository.

Scripting API

Javadoc that describes the scripting system and types of scripts that can be written.

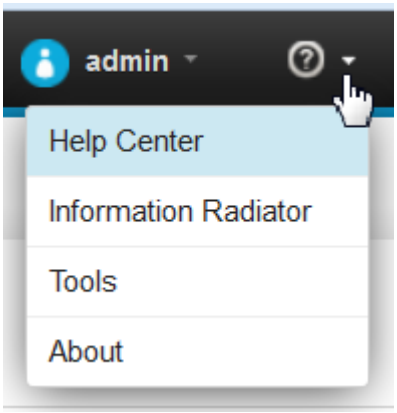
REST API

Documentation describing the REST API and how different endpoints can be used.

Build Plug-in Update Site

Run builds and download project dependencies from within an Eclipse IDE.

1. From the Build dashboard, click Help icon.



2. Click the **Help > Tools**.
3. From the Tools window, select the desired Build development tool.

Installing plug-ins

You install plug-ins by downloading the installation files and loading them into Build.

About this task

Plug-ins can be installed at any time. You do not have to restart the server after you install a plug-in. You can upgrade a plug-in by installing the new version over the old version.

1. Download the installation files for one or more plug-ins from Buildprovides plug-ins website.

   Documentation for plug-ins and the steps in the plug-ins is available on that site.

2. Load the plug-ins:
   a. On the left navigational panel, click **System > Plug-ins**.

   a. On the **Plug-ins** page, click **Browse** and select the downloaded plug-in file.
   You do not have to extract the compressed file.

   b. Click **Load**.

   c. Click **Done**.

Results

The plug-in is listed in **Loaded Plug-ins** list. After the plug-in is installed, it is available immediately.

Creating and Managing Projects

Projects run builds, manage dependencies with third-party tools, and can be used for administrative, operational, and system maintenance.

Projects often correspond to how the development process is structured in your source control management system. By using the dependency management system, projects can be composites of other projects.

Projects provide a wealth of information about the build and release cycle. Build projects create build lives every time builds run and that information is always available to you. Templates are provided for managing the dependencies and artifacts that are associated with projects. Because templates are reusable, you can reapply the same standards to similar projects.

Creating projects

You create projects by combining process and source definitions into complete build processes.

Before you begin

Before you create a project:

- Ensure that at least one agent is installed and running. .
- Configure a repository and make sure Build has access to it. See Repositories on page 330.
- Create a project template. See Creating project templates on page 384.

- Create a workflow. See Creating process definitions on page 390.
- Create a job and add it to the workflow. See Creating jobs on page 376.

1. On the left navigational panel, click **Projects > Create**.
2. In the **New Project** dialog box, type a name for the project in **Name** field.
3. Enter a description in the **Description** field.
4. Select a template from the **Template** list.
5. To add tags to the project, complete one of these actions:

   Choose from:

   - Select an tag from the **Tags** list.
   - Create a tag: type a name for the tag in the **Tags** field, and click **Add**.

   The tag is displayed in the **New Project** dialog box. Any number of tags can be assigned to the project.
6. From the **Team** list, select a team to manage the project
7. Click **Save**.

What to do next

To further define the project, see these topics for more information:

- Configuring projects: Configuring projects on page 352.
- Reviewing projects: Monitoring projects on page 369.
- Running projects: Running projects on page 370.
- Managing subscription notifications: Managing notification subscriptions on page 365.

Configuring projects

You configure a project by adding build and secondary processes to it, and manage tags and notification subscriptions

1. On the left navigational panel, click **Projects**, select a project, and then click **Configuration**.
2. **Optional:** To export the project, click **Export**, and identify the location to save the project file.

   Result

   The project is saved to an an XML file with the same name as project.
3. **Optional:** To copy the project, click **Copy**, and then name the project and click **Save**.

   Result

   The copied project is available on the **Projects** page.
4. To create a build process, click **New** in the **Build Processes** area.

   For information about build processes, see Managing build processes on page 353.
5. To create a secondary process, click **New**.

   For information about secondary processes, see Managing secondary processes on page 367.
6. To manage notification subscriptions, see Managing notification subscriptions on page 365.
7. **Optional:** To manage tags, click **Edit** and complete the following steps.
   a. Select a tag from the **Tags** list to add an existing tag to the project.
   b. Type a new tag in the **Tags** box to add a new tag to the project.

c. Click the Delete action associated with a tag to remove it from the project.
d. Click **Save**.

What to do next

To review and monitor projects, see Monitoring projects on page 369.

Managing build processes

Build processes define the input, transformational processes, and the expected output of projects

Before you begin

Before you create a build process, complete these actions:

- Create a process. See Creating process definitions on page 390.
- Create at least one job and add it to the process. See Creating jobs on page 376.

About this task

Build process input is source code that is stored in a source code manager (SCM). The actions that transform source code into the expected output are defined by processes and jobs. The results are called build artifacts.

1. On the left navigational panel, click **Projects**, and then select a project.
2. Click **Configuration**.
3. On the **Configuration** page, click **New** in the **Build Processes** area.
4. Define the process by completing these steps:

    a. Select a process template from the **Template** list.
    For information about build process templates, see Creating build process templates on page 388.

    b. Type a name for the process in the **Name** field.

    c. Select a team to manage the process from the **Team** list.

    d. Select a scheme from the **Notification Scheme** list.
    For information about notification schemes, see User notifications on page 311.

    Note: If a notification scheme is not selected for the process, notification subscriptions will not work. For information about managing notification subscriptions, see Managing notification subscriptions on page 365.

    e. Select a priority option from the **Priority** list.
    Priority options determine when processes run. Processes with a **High** priority run before processes with **Normal** or **Low** priorities. Processes with **Normal** priority run before processes with **Low** priority. When a process begins, it is not suspended or stopped when a higher-priority process is requested. **Normal** is the default option.

f. Select an option from the **Skip Pre-Processing** options.
If you select **Yes**, build preprocessing, such as checking for source changes, is skipped. The default selection is **No**. For information about preprocessing, see User notifications on page 311.

g. Click **Save**.

5. Define a source configuration by completing these steps:

a. Click **Create** in the **Source Configuration** area.

b. Using **New Source from Template** window, select a template from the **Source Template** list.
For information about creating source templates, see Creating source templates on page 387.

c. Type a name for the source configuration in the **Name** field.

d. Select a repository from the **Repository** list.

6. **Optional:** Add process properties by completing these steps:

a. Click **Create Property** in the **Properties** area.

b. Type a name for the property in the **Name** field.

c. Specify whether users can override the default value by selecting the **User May Override** option.
If selected, users are prompted to specify a value when they run the process.

d. Enter a value for the property in the **Value** field.

7. **Optional:** Define a build configuration by completing these steps:

a. Click **Create** in the **Build Configurations** area.

b. Type a name for the build configuration in the **Name** field.

c. **Optional:** Specify values for the available properties.
The properties are derived from the project templates and vary depending on the project.

d. Click **Save**.

8. Define dependencies for the build process by completing these steps:

a. Click **Dependencies**.

b. Select an option from the **Conflict Strategy** list.
This option determines what happens when a dependency conflict occurs. **Fail** means do not continue until conflicts are resolved. **Favor Old** means that the oldest build's artifacts are used. **Favor New** means that the newest build's artifacts are used. **Fail** is the default option.

c. Select **Trigger Only Dependencies** if you want dependencies to trigger only builds.
If cleared, dependencies trigger builds and set additional dependencies. Select this option when dependencies are determined by external dependency management tools, but you want to use dependencies as triggers. The default option is cleared.

9. Define a trigger by completing these steps:

a. Click **Triggers**, and then click **New Trigger**.

b. Select a trigger type from the **Type** list, and click **Select**.
The trigger types are described in the following table.

Table 12. Trigger types

Trigger type	Description
Scheduled Trigger	Scheduled triggers run on their own timers that poll the SCM tools for changes. When changes are detected, the build starts at the scheduled time. Scheduled triggers can be configured to force builds regardless of whether source changes occurred.
Repository Trigger	Repository triggers initiate builds whenever source code changes occur in the repository.
Post-Process Trigger	Post-processing triggers listen for post-processing events. When an event is detected, it can trigger other events. See Creating post-processing scripts on page 309.

c. Type a name for the trigger in the **Name** list.

d. **Optional:** If you selected **Scheduled Trigger**, select a schedule from the **Schedule** list.
For information about schedules, see Creating schedules on page 332.

e. Select **Force** to force builds when triggers occur.
If **Force** is selected, builds are triggered even if source changes are not detected.

f. Select a priority option from the **Priority** list.
Priority options determine when queued build requests run. Requests with a **High** priority run before requests with **Normal** or **Low** priorities. Requests with **Normal** priority run before requests with **Low** priority. **Normal** is the default option.

g. Selected **Enabled** to make the trigger active.

h. **Optional:** If you selected **Post-Processing Trigger**, select a process from the **Secondary Process** list.

For information about schedules, see Creating schedules on page 332.

i. Click **Save**.

What to do next

To see information about the teams that are assigned to the process and their permissions, click **Security**.

Defining source configurations

A source configuration defines a project's source type and the type of build artifacts that the project generates.

Before you begin

Before you create a build configuration, create at least one source template, see Creating source templates on page 387.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process, and click **Configuration**.
3. Define a source configuration by completing these steps:

    a. Click **Create** in the **Source Configuration** area.

    b. Using **New Source from Template** window, select a template from the **Source Template** list.
    For information about creating source templates, see Creating source templates on page 387.

    c. Type a name for the source configuration in the **Name** field.

    d. Select a repository from the **Repository** list.

4. Click **Save**.

Defining build configurations

Build configurations are collections of properties that are used to configure runtime project properties.

Before you begin

Before you create a build configuration, create the build process to which you want to attach it, see Managing build processes on page 353.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process, and click **Configuration**.
3. Define a build configuration by completing these steps:

    a. Click **Create** in the **Build Configurations** area.

    b. Type a name for the build configuration in the **Name** field.

c. **Optional:** Specify values for the available properties.
The properties are derived from the project templates and vary depending on the project.

d. Click **Save**.

Defining build properties

Build properties can be used by builds that are associated with the build process.

Before you begin

Before you create a build configuration, create the build process to which you want to attach it, see Managing build processes on page 353.

About this task

Build properties typically hold variable data for builds, such as compiler versions, or are used for an audit trail.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process, and click **Configuration**.
3. Add process properties by completing these steps:

   a. Click **Create Property** in the **Properties** area.

   b. Type a name for the property in the **Name** field.

   c. Specify whether users can override the default value by selecting the **User May Override** option.
   If selected, users are prompted to specify a value when they run the process.

   d. Enter a value for the property in the **Value** field.

4. Click **Save**.

Moving build processes

Defined processes can be moved from one project to another.

1. On the left navigational panel, click **Projects**, and then select the project that contains the process to move.
2. Click **Configuration**.
   On the **Configuration** window, under the **Build Processes** area the processes are listed.
3. Under **Actions** for the process to move, click the move icon ( ).
4. In the **Select Project** window, select the project to move the process to.
5. Click **Select**.
   On the **Projects**, the process is now under the selected project.

Defining project dependencies

Dependency configurations are used to trigger builds in related projects.

Before you begin

Before you create a build configuration, create the build process to which you want to attach it, see Managing build processes on page 353.

About this task

Projects often require outside libraries or files. The dependent project can use artifacts that child projects produce. Dependency configurations can also trigger related builds. A dependency can determine whether its dependencies must be built before it can build or whether the dependencies can trigger builds that depend on this build.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process.
3. Define dependencies for the build process by completing these steps:
    a. Click **Dependencies**.
    b. Select an option from the **Conflict Strategy** list.
    This option determines what happens when a dependency conflict occurs. **Fail** means do not continue until conflicts are resolved. **Favor Old** means that the oldest build's artifacts are used. **Favor New** means that the newest build's artifacts are used. **Fail** is the default option.
    c. Select **Trigger Only Dependencies** if you want dependencies to trigger only builds.
    If cleared, dependencies trigger builds and set additional dependencies. Select this option when dependencies are determined by external dependency management tools, but you want to use dependencies as triggers. The default option is cleared.
4. Click **Save**.

Configuring build artifacts

Build artifacts can be configured within the build template or on a build process based on the created artifact sets.

Before you begin

The artifact sets must be created before you can configure them. See, Creating artifact sets on page 327.

About this task

Configuring artifacts for a build process on templates provides for standardization among build processes. However, sometimes you require a more specific configuration for a build. By configuring artifacts on the build process you can create this specialization. The artifacts that are configured at the build level are concatenated with those that are defined at the template level and apply to the build process they are configured on. For information about creating templates, see Creating build process templates on page 388.

1. On the left navigational panel, click **Projects**.
2. Expand the project to configure by clicking the expand icon to display the project's build processes and select a build process.
3. Click the **Artifacts** tab.
   The **Artifacts** page displays the currently configured artifacts and their properties.
4. To add a new artifact configuration, click **New Artifact Config**.
   a. Select the artifact set from the associated menu. Remember, only configured artifact sets are listed.
   b. Specify the base directory in the **Base Directory** field. The directory is relative to the working directory when the Publish Artifacts step runs. To use the working directory, enter a period (.). You can use process and project properties to define the location, such as ${property: value}.
   c. Specify the artifacts to retrieve in the **Include Artifacts** field. You can use the * and ** wildcards to define the pattern. The ** wildcard indicates every directory and the* wildcard indicates every file. For example, the `dist/**/*` pattern retrieves every folder and file beneath the `dist` directory. You can also use process and project properties to define the pattern.
   d. To exclude artifacts that would otherwise be included, specify them in the **Exclude Artifacts** field. You can use process and project properties to define the pattern.
   e. Click **Save**.

Defining build triggers

Triggers can determine when builds occur. There are several types of triggers.

Before you begin

Create a build process to which you want to add a trigger.

About this task

Builds can be triggered by repository events, such as source code changes; triggered on a schedule; or triggered by post-processing events, such as completing builds in related projects.

1. On the left navigational panel, click **Projects**, and then use the expand icon associated with a project to display the project's build processes.
2. Select a build process.
3. Define a trigger by completing these steps:
   a. Click **Triggers**, and then click **New Trigger**.
   b. Select a trigger type from the **Type** list, and click **Select**.
      The trigger types are described in the following table.

Table 13. Trigger types

Trigger type	Description
Scheduled Trigger	Scheduled triggers run on their own timers that poll the SCM tools for changes. When changes are detected, the build starts at the scheduled time. Scheduled triggers can be configured to force builds regardless of whether source changes occurred.
Repository Trigger	Repository triggers initiate builds whenever source code changes occur in the repository.
Post-Process Trigger	Post-processing triggers listen for post-processing events. When an event is detected, it can trigger other events. See Creating post-processing scripts on page 309.

c. Type a name for the trigger in the **Name** list.

d. **Optional:** If you selected **Scheduled Trigger**, select a schedule from the **Schedule** list.
For information about schedules, see Creating schedules on page 332.

e. Select **Force** to force builds when triggers occur.
If **Force** is selected, builds are triggered even if source changes are not detected.

f. Select a priority option from the **Priority** list.
Priority options determine when queued build requests run. Requests with a **High** priority run before requests with **Normal** or **Low** priorities. Requests with **Normal** priority run before requests with **Low** priority. **Normal** is the default option.

g. Selected **Enabled** to make the trigger active.

h. **Optional:** If you selected **Post-Processing Trigger**, select a process from the **Secondary Process** list.
For information about schedules, see Creating schedules on page 332.

i. Click **Save**.

Monitoring build processes

You can review activities that are related to build processes, historical information that is related to the build process, and review information about individual build lives.

Before you begin

Create a build process and run one or more builds.

About this task

The **Main** tab for a build process displays information about recent build statuses and recent build lives that are generated from the build process. Graphs of related information, such as the percent of completed builds, can be displayed. Controls are available to run builds with this build process. See Running projects on page 370 for information about running builds.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process.
3. **Optional:** To see information that is related to changes that are made to the build process, click **Changes**.
   Result
   You can order the list of changes by build or by user.
4. To see information about the build lives that are created from this build process, click **My Activity**.
   Result
   Builds that are generated by your ID that use this build process are displayed.

What to do next

For information about running builds, see Running projects on page 370. For information about particular build lives, see Managing build lives on page 361.

Managing build lives

You can review current and historical information that is related to a build life.

Before you begin

Create a build process and run one or more builds.

About this task

Several tools are provided to help you manage build lives and diagnose problems.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process, and then select a build life.
   Result
   The **Main** tab for a build life displays information about the build and provides access to reports and artifacts that are generated by the build. Additionally, the agent used by the build processes is displayed.
3. **Optional:** To complete actions that are related to the build life, complete these steps.

   a. To see information that is related to the build request that generated the build life, click **View Request**.
      Result
      The **Request** page displays information about the user who requested the build and the project that is associated with it. You can access the build log and the request's context.

   b. To see information about the build's context, click **View Context**.
      Result

The **Request Context** page displays information about related process requests. If a request triggers another request by dependencies, the requests are in the same context. All requests that are created by a schedule are in the same context. For additional information about the features on the **Request Context** page, see View build context on page 363.

c. To see information about the build process's configuration, click **View Configuration**.
See Managing build processes on page 353.

d. To see information about a job, click the job in the **Process/Job/Step** list.
Result
The **Job Execution** page displays information about the processed job. You can access the output log for the job and the job template.

e. To restart the build, use the Restart icon .

f. To run a secondary process, use the Run Secondary Process icon , and select a secondary process.

g. To add a stamp the build, click **Add Stamp**, and specify the stamp.

h. To add a label to the build, click **Assign Label**, and specify the label.

i. To deactivate the build life, click **Deactivate**.
When you deactivate a build life, it is permanently disabled and its artifacts removed. A deactivated build life can never be enabled nor used in dependency relationships. A build life cannot be deactivated until all the dependency relationships are resolved.

j. To delete the build life, click **Delete**.
Deleting a build life permanently removes all traces of the build life. This action cannot be reversed.

k. To archive a build life, click **Archive**.
This action stores the build properties, such as time of build, labels, and build number, and removes artifacts from CodeStation. Dependencies are locked. Archived build lives can be unarchived.

4. **Optional:** To review information that is related to the build life, complete these actions:

a. To see reports about the build life, click **Reports**.
Result
The **Reports** page displays information about published reports and links.

b. To see the artifacts that are generated by the build life, click **Artifacts**.
Result
The **Artifacts** page displays information about the generated artifacts. You can also download and review them.

c. To see the build life's dependent relationships, click **Dependencies**.
Result
The **Dependencies** page displays information about projects the build depends on, and other projects that depend on this build.

d. To see information about source and dependency changes, click the job in the **Dependencies**.

e. To see information about errors that are related to the build, click **Issues**.
Result
The **Issues** page displays information from issue trackers and represents defects or work items that are listed in source changes.

f. To see information about tests that were done on the build, click **Tests**.
Result
The **Tests** page displays test results that are related to the build.

g. To see information about test coverage for the build, click **Coverage**.
Coverage integrations collect information on coverage tests that were run on the build.

h. To see analytic reports, click **Analytics**.
Result
The **Analytics** page displays the results of source code analysis reports.

i. To add a note to the build life, click **Notes**, and then **Add Note**, and type the note.

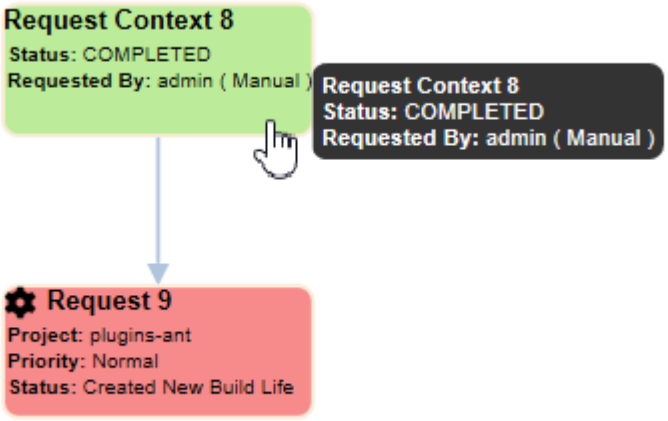
View build context

The **Request** page displays information about a build and the projects that are associated with it.

The **Request Context** page displays information about related process requests. A tree view of the build dependencies is displayed, which provides a hierarchical graphic of the dependencies. All requests that are created by a schedule are in the shown in the tree view.

Child nodes can be hidden, by clicking on the parent node. When nodes are hidden, a black border is shown on the parent node. Click the parent node to expand its child nodes.

The color of the tree nodes indicate the status of the project in the node. Additional information such as project name and date can be obtained by hovering over the node.



The following table lists the colors used to describe the status of a node.

Table 14. Status details

Color	Status
Green	Completed
Yellow	Running
Grey	The project is queued, waiting for an agent or dependency
Red	Failed
Blue	Suspended

Searching for build lives

Use the **Search** window to locate build lives or miscellaneous jobs using a set of specific criteria.

About this task

You can search using a build, request, process, status, or label criterion to search for build lives. To search for a job, use the **Misc Job** tab.

1. On the left navigational panel, click **Search**.
2. Select the tab for the type of criteria that you want to use for your search.
   Choose from:

- To search by build lives, click **Build**. Filter the search based on project name, build ID, latest status, or latest stamp. The latest stamp filter supports using the asterisk wildcard (*) character to match any sequence of characters.
- To search by build request status, click **Request**. Filter the search based on request status and project name. The project name filter supports using the asterisk wildcard (*) character to match any sequence of characters.
- To search by process status, click **Process**. Filter the search based on process status and project name. The project name filter supports using the asterisk wildcard (*) character to match any sequence of characters.
- To search by status, click **Status**. Filter the search based on build life status and project name. The project name filter supports using the asterisk wildcard (*) character to match any sequence of characters.
- To search by label, click **Label**. Filter the search based on a build life label name and project name.

  In addition to searching, you can also create a label, and assign and remove a label from build lives.
  - To assign a label, select the desired builds by using the check boxes on the left side of the table. Select a label from the menu and click **Assign**.
  - To remove a label, select the desired builds using the check boxes on the left side of the table. Select a label from the menu and click **Remove Label**. The label is removed from the selected build lives; it is not removed from the server. To remove a label from the server, see Managing Labels on page 329.
  - To create a label, click **Add Additional Label** and type in the new label name. After the label is created, it can be assigned to a build life by clicking **Assign**.
- To search for jobs using a project name, click **Misc Jobs** and limit the data using the filters on the **Project** field. Type a number in the **Results Per Page** field to limit the number of displayed results. The default value is 40.

Managing notification subscriptions

You can receive email notifications whenever one of your projects run a build.

Before you begin

Make sure that you configure a notification scheme before you manage subscription notifications. For information about notification schemes, see Creating notification schemes on page 312.

About this task

Notifications are triggered based on build statuses that you specify. Notification might be sent whenever a build fails, for example. The notification scheme that is associated with the process determines who receives email notifications. If a notification scheme is not associated with a process, then no notifications are sent regardless of the notification subscription settings

You can use the notification subscription feature to unsubscribe to notifications that you would otherwise receive. You can elect to unsubscribe from all notifications for a project or unsubscribe from specific processes. You can also configure your user account to receive no notifications. If you unsubscribe from notifications, you will not receive any regardless of the notification subscription settings.

1. On the left navigational panel, click **Projects**, and then click the name of the project.
2. In the **Main** page of the project, click **Configuration**.
3. In the **Notification Subscription** field, click the name of the currently selected option. Click **None** if no option is selected.

   For example, if the currently selected option is **Default**, click .

   Result

   The **Notification Subscription** dialog box is displayed. The **Notification Subscription** dialog box displays lists of options for the project itself and each process that is associated with the project. You can also manage subscriptions on the individual process **Configuration** pages. The procedure is the same regardless of where you access the feature.
4. Select a trigger option from the list that is associated with the project or process.
   You can change options independently of each other. The options are.

Table 15. Notification subscription options

Option	Description
Default	Receive notifications based on the settings of the scheme that is associated with the process.
Failure	Receive notifications only when the process fails.
None	Do not receive notifications.
Status Change	Receive notification only when the build status of the process changes.
Success	Receive notifications only when the process succeeds.
Success and Failure	Receive notifications whenever the process succeeds or fails.

Result

Notifications are sent to the users specified by the notification scheme whenever the triggering condition is met. Use this table to determine when notifications are sent when the project setting is different from the setting for individual processes. If you configure your user account to unsubscribe from notifications, you will not receive any notifications regardless of the subscription settings. It bears repeating that if a notification scheme is not configured for a process, no notifications are sent regardless of the notification subscription settings.

Table 16. Notification subscription settings

Project setting	Process setting	When notifications are sent
Success	Default	Success
Success	None	Not sent

Project setting	Process setting	When notifications are sent
None	Success	Not sent
Default	Success	Success
Default	Default	Default
Failure	Success	Success
Default	Success/Failure	Success or failure
None	Default	Not sent

5. **Optional:** Click the **Delete**, , action that is associated with the currently selected option to delete the subscription setting that is associated with the project or process.

Managing secondary processes

Secondary processes manipulate or operate on the artifacts that build processes generate.

Before you begin

Before you create a secondary process, complete these actions:

- Create a secondary process. See Creating process definitions on page 390.
- Create at least one job, and add it to the process. See Creating jobs on page 376.

About this task

The artifacts that build processes generate are stored in CodeStation and are available for more processing, which is called secondary processing. Most secondary processes run tests on build artifacts.

1. On the left navigational panel, click **Projects**, and then use the expand icon that is associated with a project to display the project's build processes.
2. Select a build process, and click **Configuration**.
3. Click **New** in the **Secondary Process** area.
4. Define the process by completing these steps.
    a. Select a process template from the **Template** list.
    For information about build process templates, see Creating build process templates on page 388.
    b. Type a name for the process in the **Name** field.
    c. Select a team to manage the process from the **Team** list.
    d. Select a scheme from the **Notification Scheme** list.
    For information about notification schemes, see User notifications on page 311.

e. Select a priority option from the **Priority** list.
Priority options determine when processes run. Processes with a **High** priority run before processes with **Normal** or **Low** priorities. Processes with **Normal** priority run before processes with **Low** priority. After a process begins, it is not suspended or stopped when a higher-priority process is requested. **Normal** is the default option.

f. Click **Save**.

5. Add process properties by completing these steps:

a. Click **Create Property** in the **Properties** area.

b. Type a name for the property in the **Name** field.

c. Specify whether users can override the default value by selecting the **User May Override** option.
If selected, users are prompted to specify a value when they run the process.

d. Enter a value for the property in the **Value** field.

6. Define a trigger by completing these steps:

a. Click **Triggers**, and then click **New Trigger**.

b. Select a trigger type from the **Type** list, and click **Select**.
The trigger types are described in the following table.

Table 17. Trigger types

Trigger type	Description
Scheduled Trigger	Scheduled triggers run on their own timers that poll the SCM tools for changes. When changes are detected, the build starts at the scheduled time. Scheduled triggers can be configured to force builds regardless of whether source changes occurred.
Repository Trigger	Repository triggers initiate builds whenever source code changes occur in the repository.
Post-Process Trigger	Post-processing triggers listen for post-processing events. When an event is detected, it can trigger other events. See Creating post-processing scripts on page 309.

c. Type a name for the trigger in the **Name** list.

d. **Optional:** If you selected **Scheduled Trigger**, select a schedule from the **Schedule** list.
For information about schedules, see Creating schedules on page 332.

e. Select **Force** to force builds when triggers occur.
   If **Force** is selected, builds are triggered even if source changes are not detected.

f. Select a priority option from the **Priority** list.
   Priority options determine when queued build requests run. Requests with a **High** priority run before requests with **Normal** or **Low** priorities. Requests with **Normal** priority run before requests with **Low** priority. **Normal** is the default option.

g. Selected **Enabled** to make the trigger active.

h. **Optional:** If you selected **Post-Processing Trigger**, select a process from the **Secondary Process** list.
   For information about schedules, see Creating schedules on page 332.

i. Click **Save**.

What to do next

To see information about the teams that are assigned to the process and their permissions, click **Security**.

Monitoring projects

You can review activities that are related to projects and project build processes, and create and import projects.

About this task

The **Projects** tab displays information about projects, recent build lives that are generated from the projects. Graphs of related information, such as the daily success rate, can be displayed. Controls are available to create and import projects.

1. On the left navigational panel, click **Projects**.
   Result
   The **Projects** tab lists all projects and their status.
2. To filter the list of projects, enter search criteria in the **Project** field and click **Filter**.
3. To see the build processes that are associated with a project, click the expand icon that is associated with the project.
4. **Optional:** To see information about a build, click the View Build icon that is associated with the build.
   For information about build lives, see Managing build lives on page 361.
5. To create a project, click **Create**.
   For information about creating projects, see Creating projects on page 351.
6. To import a project, click **Import**, and complete these steps:
   Choose from:
   - To import a project file, click **Browse** and select the file; then, click **Import**.
   - Paste the project definition in the text box, and then click **Import**. The pasted file must be an XML file that was previously exported from Build.
7. To view inactive projects, click **View Inactive**.
   Result

Inactive projects are displayed.

8. To see information about the build lives that were created from the listed projects, click **My Activity**.
   Result
   Builds that were generated by your user ID are displayed. For information about build lives, see Managing build lives on page 361.

What to do next

For information about running builds, see Running projects on page 370. For information about particular build lives, see Managing build lives on page 361.

Creating and Managing Projects

Projects run builds, manage dependencies with third-party tools, and can be used for administrative, operational, and system maintenance.

Projects often correspond to how the development process is structured in your source control management system. By using the dependency management system, projects can be composites of other projects.

Projects provide a wealth of information about the build and release cycle. Build projects create build lives every time builds run and that information is always available to you. Templates are provided for managing the dependencies and artifacts that are associated with projects. Because templates are reusable, you can reapply the same standards to similar projects.

Running projects

When you run a project, you select and run a particular build process that is associated with the project.

1. On the left navigational panel, click **Projects**.
2. Click the expand icon that is associated with a project, and then select the build process to run.
3. On the build process page, select **Force** if you want to force the build.
4. Select **Delay Build** if you want to delay the build.
5. **Optional:** Select a configuration from the **Build Configuration** list.
   For information about build configurations, see Defining build configurations on page 356. The default selection is **None**.
6. Click **Build**.
   Result
   Information about the build is displayed on the **My Activity** page for the build life. Initially, the status is `Running`. A successful build has the status of **Complete**. An unsuccessful build has the status of **Failed**.

What to do next

For information about build lives, see Managing build lives on page 361.

Running preflight builds

You can run test builds in your production build environment without impacting your production environment, using a preflight build. Preflight builds allow you to build code that has not been merged into your production code to determine if the build completes correctly.

About this task

A preflight build is like a regular build life, allowing you to test a build before committing code changes to the source control. The preflight build includes a clean-up service that deletes preflight build older than thirty days. This prevents the accumulation of test builds using server storage.

A preflight build cannot be specified as a dependency for either a regular or another preflight build. Because it can never be a dependency for any other build, it is always safe to delete.

To use this feature, you must download the Preflight client on any computer that can communicate with the Build server.

1. In the Build user interface, click **Help > Tools** on the taskbar and select the Preflight package to download the preflight client.
2. Go to the location of the package that you downloaded and decompress the package. The build executable is in the `/bin` directory.
3. Start a preflight build using a command line interface as shown in the example below. In the example, a preflight build is started for the project with an ID of 1423 and build process number 52. The user ID and password are included in the command to authentication with the build server.

```
preflight build --url http://localhost:8080 --local-server-host localhost -p 1423 -bp 52
-D workspace.revision.6df88f60-0ed2-4da2-8fcc-12fd779d8bfa=cf5cf4a6a185b523fa2d80cf3356981a759a441c
--user PasswordIsAuthToken --password 31e3dcc9-96d5-4640-98f6-0583a6928355
```

--url

Fully-qualified URL of the Build server

--local-server-host

Local name of the Build server. The value is usually `localhost`.

-p

The ID of the project.

-bp

The ID of the build process.

-D

Properties to include with the build. Specify the **workspace.revision.uniqueID** property for the build to use a specific code commit. This property contains information from the **Populate Workspace** step to identify the correct code to use. If a value for the property has not been set, the latest commit is used and the property is set to the latest commit unique ID value.

--user

The user ID to log into the Build server.

--password

The password associated with the user ID.

What to do next

After the build completes, view the output information to verify that the build completed successfully. Below is sample of a Preflight build output.

```
Request URL:
        http://localhost:8080/tasks/project/BuildRequestTasks/viewBuildRequest?buildRequestId=463Request
 Status:
        SubmittedRequest Status:
        BUILD_LIFE_CREATED
        Buildlife URL: http://localhost:8080/tasks/project/BuildLifeTasks/viewBuildLife?buildLif eId=386
        Build Status:  RUNNING
        Build Status: FAILED
```

Notice that the **Build Status** field indicates that the preflight build failed. The error should be investigated and fixed before merging into the production code. The link in the **Buildlife URL** field provides additional information about the failure.

You can continue using the preflight build testing iterations of the code until a successful build occurs.

CodeStation projects

CodeStation projects model external tools or libraries as Build projects.

With CodeStation projects, you can use third-party tools as dependencies, and test and track their status within an application. For example, if your application requires Java 5, you can create a CodeStation project and create an artifact set that contains the Java 5 files. Projects that require Java 5 can use the artifact set as a dependency. CodeStation projects can support multiple versions of the libraries in an artifact set. When the dependency is called during a build, your settings determine which version to use. For example, one process of your project can depend on library version 1 and another component can depend on version 2 of the same CodeStation project.

It is important to note that CodeStation projects cannot depend on other projects.

Creating CodeStation projects

CodeStation projects model external tools or libraries as Build projects.

1. On the left navigational panel, click **CodeStation > Create CodeStation Project**.
2. Type a name for the CodeStation project in the **Name** field.
3. Click **Save**.
4. **Optional:** Add properties to the project by completing the following steps:

a. Click **Properties** and click **New Property**.

b. Type a name for the property in the **Name** field.

c. Enter a value for the property in the **Value** field.
Property values are referenced by using this syntax: `${p:name}`.

5. Click **Save**.

What to do next

Create artifact sets by loading files into the project. See, Creating CodeStation project artifact set versions on page 373. To see the teams assigned to the project and their permissions, click **Security**.

Creating CodeStation project artifact set versions

A CodeStation artifact set is a collection of files that were not created by Build. CodeStation artifact sets are used as dependencies for other projects.

Before you begin

Create the CodeStation project for which you want to create an artifact set version.

About this task

Each time that you load files into a CodeStation artifact set, an artifact set version is created. A CodeStation artifact set can have any number of versions.

1. On the left navigational panel, click **CodeStation**, then select a CodeStation project.
2. Click **New** in the **Versions** area.
3. Type an identifier in the **Stamp** field.
A stamp typically uses an algorithm with an incremental property such as `${+property:propname}`. A stamp can also use constant, such as `myStamp`.
4. Click **Save**.
5. Load files into the version by completing the following steps.

a. Click **Artifacts**.

b. Click **Upload File**, or **Upload Many Files**.
To upload several files, first create a compressed file that contains the files.

c. Select an option from the **Artifact Set** list.
This option assigns a name to the artifact set.

d. Specify the directory where the files are in the **Directory** field.

e. Use **Browse** and select the file.
If you selected **Upload Many Files**, the file must be a compressed-type file.

f. Click **Upload Files**.

6. **Optional:** Define a property for the artifact set version by completing the following steps.
    a. Click **Properties** and click **New Property**.
    b. Type a name for the property in the **Name** field.
    c. Enter a value for the property in the **Value** field.
    Property values are referenced by using this syntax: `${p:name}`. These properties cannot have empty values.
    d. Click **Save**.
7. Add statuses to the artifact set version by completing the following steps.
    a. Click **Status** and click **Add Status**.
    b. Select a status from the **Status** list.
    For information about creating statuses, see Creating statuses on page 334.
    c. Click **Add**.

What to do next

To see projects that depend on the CodeStation project version, click **Dependencies**.

Build pipelines

A build pipeline defines the automated process that transforms source code into deployable artifacts. It specifies the steps required to compile, test, validate, scan, and package the code.

In Build, you can define a pipeline as a **Process** and map each stage of the pipeline to a **Job**.

Pipeline stages as Jobs

A pipeline includes multiple stages. In Build, you can implement each stage as a Job.

A typical build pipeline consists of the following stages:

Table 18. Typical pipeline stages

Stages	Tasks
Build	Compile source code and generate binaries.
Test	Run automated unit and integration tests.
Scan	Perform security, compliance, or quality scans.
Package	Create deployable artifacts.
Publish	Upload artifacts to a repository.

You can create reusable Job templates for the above stages and add them in a Process to standardize execution across pipelines.

Multistage pipeline

Build supports multistage pipeline so that you can define multiple Job templates and combine them in a Process.

To configure a multistage pipeline, complete the following steps:

1. Create Job templates for each required stage.
2. Add the Jobs to a Process definition in the required execution order.
3. Configure dependencies between Jobs to control execution flow.

The system runs the Jobs sequentially according to the defined order. Each Job represents a distinct stage. Downstream Jobs can consume artifacts produced by upstream Jobs. If a Job fails, the system stops or controls subsequent execution based on your configuration.

This multistage pipeline design enforces quality gates and ensures controlled progression.

Parallel pipeline

You can configure Jobs within a Process to run in parallel when no dependency exists between them.

To configure parallel execution, complete the following steps:

1. Add multiple Jobs to the same Process stage.
2. Do not define dependency relationships between those Jobs.
3. Ensure sufficient system resources to support concurrent execution.

The system executes the Jobs simultaneously. Parallel execution reduces overall build time and optimizes resource usage.

Typical parallel execution scenarios are as follows:

- Running multiple test suites simultaneously.
- Running security scans and code quality analysis concurrently with testing.

For detailed procedures on creating Job templates and Process definitions, refer to Creating Jobs and Processes on page 375.

Creating Jobs and Processes

Jobs are the core building blocks of Build. Process definitions assemble jobs into the basic unit of automation. A process definition determines how, when, and where jobs run.

Creating jobs

A job is a series of steps that constitute a process, usually a build. Job steps are processed one at a time in the order you specify.

About this task

The steps that are used to create a job are taken from plug-ins. You can think of a plug-in step as distinct pieces of automation. By combining steps from various plug-ins, you can create fully automated processes. The number of steps that are provided by a particular plug-in can vary, and each step has a variable number of properties. Property values can be supplied when defining a job or enter values at run time.

1. On the left navigational panel, click **Template > Jobs**.
2. Click **Create**.
3. In the **Create a Job** dialog box, type a name for the job in the **Name** field.
4. From the **Teams** list, select a team to manage the workflow.
5. Click **Save**.
   The job is created.
6. Click the **Steps** tab to add steps to the job on the **Jobs** details page.

    a. Click **Create Step**.
    The **Step Types** tab and a list of available steps displays. Installed plug-ins provide the available steps.

    b. Select a step.
    Expand the plug-in name to see individual steps.

    c. Click the step name.
    Result
    A tab opens with the properties associated with the step. Both properties that are specific to the selected step and general properties that are available for every step are displayed.

    d. Enter the property values.
    To see the properties that are available for every step, click **Show Additional Options**. For information about the properties common to all steps, see Configuring properties common to all steps on page 377.
    Result
    The step is added to the list of steps. For environmental properties, click **Show Environmental Variables**.

   Job steps are run in list order from the top of the list.
7. Click **Save**.

What to do next

You can insert steps before or after other steps. To insert a step before another step, click the Insert Before icon for the step before which you want to insert a step. To insert a step after another step, use the associated Insert After icon . You can use the Reorder icon to drag steps to rearrange their order.

Related information

Adding, copying, and arranging job steps on page 378

Disabling and deleting job steps on page 383

Configuring properties common to all steps

A number of properties are common to all steps.

About this task

You can specify common property values when you create the step or edit them after the step is saved. The default values for these common properties are sufficient for most situations.

1. On the left navigational panel, click **Template > Jobs**.
2. Select the job with the step that you want to edit.
3. Use the Edit icon associated with the step.
4. You can edit the displayed values of the step.
5. Click **Show Additional Options**.
   Use the following table to specify step property values.

Table 19. Step common properties

Property	Description
Is Active	Select **No** to temporarily deactivate this step without having to delete it. The default value is **Yes**.
Run in Parallel	The step runs at the same time as adjacent steps that are also parallel. The default value is **No**.
Run in Preflight	Select **No** if you do not want this step to run in preflight builds. The default value is **Yes**.
Run in Preflight Only	Select **Yes** if you want to run this step only in preflight builds. The default value is **No**.
Pre-Condition Script	Select the precondition script that must be successful before the step can run. The default value is **All Prior Success**.

Property	Description
Ignore Failures	Select **Yes** if the step status does not affect job continuation or status. Selecting **Yes** means the success or failure of this step is irrelevant. The default value is **No**.
Post-Processing Script	Specify the script that is used to determine whether the step commands fail or succeeds. To create a script, click **New Script**. For information about post-processing scripts, see The post-processing element on page 416.
Timeout	Enter the number of minutes after the step starts when the step is considered timed out and aborted. When a step times out, it is marked as failed and job execution continues based on that outcome. The default value is 0, which means that the step never times out.

6. Click **Save**.

Adding, copying, and arranging job steps

The list of steps associated with a job are listed in the **Steps** window of the job. There add steps several ways.

About this task

After you add an initial step by clicking **Create Step**, you can add additional steps using one of these methods:

- To add a step, click **Create Step**
- To insert a step before another step, click the Insert Before icon ( ) for the step before which you want to insert a step. The **Step Types** window opens for you to select from the available steps. Installed plug-ins provide the available steps. Click the a step name and complete the properties that are associated with the step.
- To insert a step after another step, use the associated Insert After icon ( ). The **Step Types** window opens for you to select from the available steps. Installed plug-ins provide the available steps. Click a step name and complete the properties that are associated with the step.
- To copy a task, click the copy icon ( ). The **Copy Step** window opens.

Creating Job templates using the Job as Code plugin

Use the Job as Code (JAC) plugin to create build job templates by defining build workflows in JSON or YAML configuration files stored in your source code repository.

Before you begin

Note: This topic applies to Build version 7.2.0.4 or above, for lower versions see Creating Job templates using the Job as Code plugin on page 380.

You must have completed the following tasks:

- Install the latest Job as Code plugin from the plugin portal.
- Ensure that a JSON or YAML configuration file, such as `builder.json`, exists in the source code repository.
- Verify that the source code repository is configured and accessible from the DevOps Build environment.

About this task

The Job as Code (JAC) plugin enables you to create and maintain build job templates as configuration files instead of manually configuring build steps through the DevOps Build user interface. By storing workflow definitions alongside source code in Git, JAC provides a single source of truth for build automation.

JAC supports configuration files stored anywhere within a source code repository. During build execution, the plugin retrieves the configuration file from the branch that triggered the build and dynamically generates the workflow defined in the file.

A configuration file can define steps such as:

- Create stamp
- Populate workspace
- Change log
- Shell script
- Status assignment
- Cleanup

Using JAC provides the following benefits:

- **Version control:** Build workflow definitions are stored with the source code and tracked through Git.
- **Self-service configuration:** Developers can modify build pipelines by updating configuration files without requiring administrative access to the CI tool.
- **Branch-specific workflows:** Different branches can use their own JSON or YAML configuration files, allowing build steps to be customized or skipped for specific branches.
- **Consistency:** Build processes are defined once and executed consistently across environments.

1. On the left navigational panel, click **Template > Jobs**.
2. Click **Create**.
3. In the **Create a Job** dialog box, type a name for the job in the **Name** field.
4. From the **Teams** list, select a team to manage the workflow.
5. Click **Save**.
6. Click the **Steps** tab to add steps to the job.

a. Click **Create Step**.
   The **Step Types** tab and a list of available steps displays. Installed plug-ins provide the available steps.

b. Select **Job as Code**.

c. Click **Configure Job as Code**.

d. In the **Name** field, enter a name for the job template.

e. In the **Config File Path** field, specify the relative path to the JSON or YAML configuration file in the repository.
   Example: `ecologic-payments/builder.json`

f. Optional: Specify `${p:triggered_branch}` to automatically retrieve the configuration file from the branch that triggered the build.
   Ensure that the configuration file name and relative path remain the same across branches.

7. Click **Save**.

Results

The job template is created and uses the workflow defined in the specified JSON or YAML configuration file. When a push or pull request triggers a build, the system dynamically generates and executes the configured workflow.

Creating Job templates using the Job as Code plugin

By using the Job as Code plug-in, you can configure a job step as a JSON or YAML file.

Before you begin

Note: This topic applies to Build version 7.2.0.3 or below, for higher versions see Creating Job templates using the Job as Code plugin on page 378.

Ensure that the Job as Code plugin is installed.

1. Select the job to add the step to, and then add a step.
   See Adding, copying, and arranging job steps on page 378 for information about adding a step.
2. In the **Step Type** window, expand **Job as Code** and click **Configure Job as Code**.
3. Enter the properties for the step.
   The following are required properties.

   Name

   The name of the JSON or YAML file.

   Job as Code File Path

   The location of the JSON or YAML file.

4. Click **Save**.

Layout of Job as Code configuration file

Using the Job-as-Code plug-in, you can configure a job step as a JSON or YAML file.

With this approach you are able to track the changes done to the job steps by maintaining the configuration file by version in an SCM repository. The configuration file must be saved on the same workstation that the Build server is installed on.

Layout of the configuration file

The configuration can be file either in JSON or YAML format. The file can contain more or more steps. Nested steps are not supported.

- If using a JSON file, identify the beginning of the steps with `"steps";`. Each step is separated by a comma. The following example, shows the layout of the JSON file.

```
{
  "steps";
  [
   {step 1},
   {step 2},
    ....
   {step n}
  ]
}
```

- If using a YAML file, identify the beginning of the steps with `steps:`. Each step begins with a dash. The following example shows the layout of the YAML file.

```
  steps:
   -step 1
   -step 2
    ....
  - step n
```

Each step contains properties that define the task.

name

The name of the step. This property is required.

stepType

The type of step. This property is required.

description

A description of the file.

workDirOffset

The working directory to use when this command runs. Specify the directory relative to current working directory.

envVars

Environment variables in format: *variable_name=value* pair.

active

Specify to temporarily deactivate the step. The default value is for the step to be active.

runInParallel

Specify to run this step at the same time as other steps marked as parallel. The default value is do not run in parallel.

runInPreflight

Specify to run this process only in preflight processes. The default is to run in preflight processes.

runInPreflightOnly

Specify to run this step only in preflight processes.

preCondition

The pre-condition script which must pass before the step is run.

ignoreFailures

Specify for this step to not affect other steps to run or determine the status of the job. The default value is the step does not affect the job or other steps.

Post-processing script

A script for determining when commands of this step should count as fail or succeed.

timeout

The number of minutes after the step starts, before it is marked as timed out and aborted. The default value is 0 meaning the step does not timeout.

properties

A list of properties that are based on the configured step type. Specify the properties as a property name value pair.

The following example shows the JSON file layout with properties:

```
[
{
  "name":"step_name",
  "stepType":"step_type",
  "description":null,
  "workDirOffset":"",
  "envVars":"",
  "active":"true",
  "runInParallel": false,
  "runInPreflight":true,
  "runInPreflightOnly":false,
  "preCondition":"",
  "ignoreFailures":"false",
```

```
  "postProcessScript":"",
  "timeout":"0",
  "properties":
  [
    {"key":"", "value":""}
```

The following example shows the YAML file with properties:

```
steps:
  -name: step_name
  stepType: step_type
  description: null
  workDirOffset: ""
  envVars: null
  active: true
  runInParallel: false
  runInPreflight: true
  runInPreflightOnly: false
  preCondition: null
  ignoreFailures:false
  postProcessScript:null
  timeout":0
  properties:
    -key: ""
     value: ""
```

Disabling and deleting job steps

You can remove a step from a job process by either disabling or deleting the step.

1. On the left navigational panel, click **Template > Jobs**.
2. Click the job to work with.
3. Click **Steps.**
4. To the right of the step to remove, complete one of these actions.
   Choose from:
   - Click the disable icon ( ). The job step is removed only from the process. To add it back click the Active icon ( ) .
   - Click the delete icon ( ) to delete the job step.

Managing Templates

Project templates are reusable plans that map build processes.

A template defines a generic configuration by defining properties that are given values by the objects created from them. For instance, a template might always invoke a particular type of build script but the name or location of the script can vary by project.

Creating project templates

When you create project template, you identify the agent pool that it uses, and configure pre-processing options. You can add project-level properties to a project.

1. On the left navigational panel, click **Templates > Create**.
2. Using the **Create a Project Template** dialog, type a name for the template in the **Name** field.
3. Select a team to manage the template from the **Team** list.
   Until you create your own teams, the System Team is available. For information about creating teams, see Security teams on page 320
4. Click **Save**.
   Result
   The **Build Pre-Processing** tab is displayed. To complete the template, you must define the required fields on this page.
5. Type a number of seconds to wait between builds in the **Time Period** field.

   The quiet period determines the amount of time that must pass after changes are made to the source before another build can begin. The value represents seconds.
6. Type a number of seconds to wait between merge triggers in the **Merge Period** field.
   The merge period is the amount of time to wait after a file is added to the source repository before another build can begin. The value represents seconds.
7. Specify criteria that are used to search for source and dependency changes.

   To use a label, type a property or a property value. To specify a property, use the property syntax, such as `${p:my.property}`. To specify a property value, type the value, such as `my property value`.

   To use a stamp, enter the stamp value or a pattern. Patterns can contain '?' for a single character, and '*' any sequence of characters. For example, `1.?` matches 1.2 but not 1.2.0; `1.*` matches both 1.2 and 1.2.0.

   Use the plus icon to enter additional values. Use the drag-and-drop icon to reorder values. If multiple values are specified, they are searched in order, top to bottom.
8. Specify an agent pool:

   a. To pick a specific pool, click **Agent Pool Selection**, then select the pool from the list.
      The job runs on the first available agent in the selected pool.

   b. To select the pool by property, click **Agent Pool Property**, then type the property, such as `${p:my.property}`.
9. To clean the working directory before processing begins, select **Yes** for the **Should Cleanup** option; otherwise, select **No**.
10. To use a precondition script for steps using this template, select the script from the **Step Pre-Condition Script** list.
    For information about step precondition scripts, see Creating step-precondition scripts on page 310
11. Click **Save**.

What to do next

After creating the project template, use the **Properties** tab to define template properties. For information, see Creating project template properties on page 386.

Configuring project template build pre-processing properties

The build pre-processing options define agent pools, and repository triggers among other things.

1. On the left navigational panel, click **Templates**, then click a project template.
2. Click **Build Pre-Processing**.
3. Click the Edit icon .
4. Type a number of seconds to wait between builds in the **Time Period** field.

   The quiet period determines the amount of time that must pass after changes are made to the source before another build can begin. The value represents seconds.
5. Type a number of seconds to wait between merge triggers in the **Merge Period** field.
   The merge period is the amount of time to wait after a file is added to the source repository before another build can begin. The value represents seconds.
6. Specify criteria that are used to search for source and dependency changes.

   To use a label, type a property or a property value. To specify a property, use the property syntax, such as `${p:my.property}`. To specify a property value, type the value, such as `my property value`.

   To use a stamp, enter the stamp value or a pattern. Patterns can contain '?' for a single character, and '*' any sequence of characters. For example, `1.?` matches 1.2 but not 1.2.0; `1.*` matches both 1.2 and 1.2.0.

   Use the plus icon to enter additional values. Use the drag-and-drop icon to reorder values. If multiple values are specified, they are searched in order, top to bottom.
7. Specify an agent pool:

   a. To pick a specific pool, click **Agent Pool Selection**, then select the pool from the list.
      The job runs on the first available agent in the selected pool.

   b. To select the pool by property, click **Agent Pool Property**, then type the property, such as `${p:my.property}`.
8. To clean the working directory before processing begins, select **Yes** for the **Should Cleanup** option; otherwise, select **No**.
9. To use a precondition script for steps using this template, select the script from the **Step Pre-Condition Script** list.
   For information about step precondition scripts, see Creating step-precondition scripts on page 310
10. Click **Save**.

Creating project template properties

Properties can be used to manage variables passed into commands, agent filters, and custom stamping algorithm templates.

About this task

Properties use the syntax `${property:[property_name]}` to pass property values. You can nest properties to create more specific output. Create a nested property by combining multiple properties. For example, if you want to pass the project name using properties: `project_name=project_1.0.0_appA` and the version number is passed in the `$version` property. If the nested property is defined as follows:

```
${p:project_${p:version\}_name}
```

and the value for the version property is 1.0.0, the property first resolves to

```
${p:project_1.0.0_name}
```

which resolves to the value: `project_1.0.0_appA`. There is no limitation on the number of nested properties. Wildcards are not supported. A "\" is required before closing each nested property.

From the **Properties** window, you can edit defined properties using the edit icon ( ) and move their position using the drag and drop icon ( ).

1. From the Build dashboard, click **Templates**, then click a project template.
2. Click the **Properties** tab.
3. Click **Create Property**.
4. In the **Create New Property** window, type a name for the property in the **Name** field.
   The property can be referenced by using the property syntax: `${p:property_name}`.
5. Select a display type from the **Display Type** list.
   The display type determines how the property is presented to users in the user interface. Based on the type you select and the **Values** field value selected, additional configuration fields are displayed. For most types, the **Default Value** field is displayed where you can specify a default value.
   Select one of the following types:
   - **Agent Pool** provides a list of agent pools. If selected, the **Agent Pool** field is displayed, where you can specify a default agent pool.
   - **Checkbox** provides a check box. If the box is checked, the value is true; otherwise, the value is false.
   - **Integration Plug-in** provides a list of integration plug-ins. Select the type of plug-in from the **Plug-in type** field.
   - **Multi-select** provides a list of values from which users can select multiple values.
   - **Select** provides a list of values from which users can select a single value.
   - **Text** provides a text-input field.
   - **Text (secure)** provides a text-input field in which user input is replaced by generic symbols.
   - **Text Area** provides a text-input field in which users can enter a large amount of text.
6. Specify the rules for the selected display type in the **Value** field.
   Depending on the selected display type, some value-related fields might not be displayed.
   Choose from:

- If you want to specify the property rules now during design time, click **Defined** and configure the following fields.
  - If you want users to specify a value for the property, click **Value Required**.
  - Enter all possible values for this property in the **Allowed Values** text area. Put each value on a separate line.
  - Optionally, enter a default value for the property in the **Default Value** field.
- If you want the property values to be generated by a preprocessing script, click **Scripted** and configure the following fields.
  - If you want users to specify a value for the property, click **Value Required**.
  - Enter a BeanShell script that generates property values in the **Allowed Values Script** text area. The script is passed the following implicit variables: `project`, `workflow`, `buildLife`. Put each value on a separate line. The script should return:
    - String value for text, text area, text (secure), and select types.
    - Boolean for the check box type.
    - String array or collection for the multi-select type.
    - The agent pool name for the agent pools type.
    - Null if the field is hidden for all property types.
  - Optionally, enter a BeanShell script that generates a default value in the **Default Value Script** text area.
- If you want the property values to be generated by a job step, click **Job Execution** and configure the following fields.
  - If you want users to specify a value for the property, click **Value Required**.
  - Select an agent pool to run the job from the **Agent Pool** list.
  - Select the job that sets the property value from the **Job** list.

7. Complete the configuration fields based on the type selected.
8. Click **Save**.

Creating source templates

Source templates define the locations of the source materials that are managed by project templates.

Before you begin

Before you create a source templates, create the project template to which you want to attach the source template. Additionally, install the source plug-in that you require if it is not already installed. For information about installing plug-ins, see Installing plug-ins on page 351.

About this task

Source templates define the location of the source material for project templates.

1. On the left navigational panel, click **Templates**, then select the project template to which you want to attach the template.
2. In the **Source** area, click **Create**.
3. Using the **New Source Configuration Template** page, select the source plug-in from the **Source Type** list, then click **Select**.

Initially, the File System source plug-in is available.

Note: The fields that are used to configure a source template can vary depending on the source type selected. The following steps describe configuring a source template with the File System source plug-in. See the documentation for your specific source plug-in.

4. Type a name for the source template in the **Name** field.
5. Select a script from the **Working Directory Script** list.

   The selected script defines the path to the working directory for the source template. Initially, there are two scripts available: **Default Workflow Directory**, and **Unique Job Directory**. The Default Workflow Directory script uses the default directory of the agent that does the work, which is `{agent}/var/work/project/{project}/{workflow}`. The Unique Job Directory script uses a separate directory for each job: `{agent}/var/work/job/{job_id}`.
6. Optional. Create your own working directory script:

   a. Click **New Script**, then select the pool from the list.
      The job runs on the first available agent in the selected pool.

   b. Type a name for the script in the **Name**, then type the property, such as `${variable}.`

   c. Enter the script in the **Script** area.
      The script can include variables that use the property syntax: `${variable}`, and BeanShell scriptlets `${bsh:...}`.

   d. Click **Save**.
      User-created scripts become available the next time the list is accessed.
7. To use an offset from the directory that is specified by the working directory script, enter it in the **Directory Offset** field.
   To specify the current working directory, use the period character '.'.
8. Optional. Specify a filter to limit the source files that are used in the **File Filters** area. To use a precondition script for steps that use this template, select the script from the **Step Pre-Condition Script** list.
   Put each filter condition on a separate line. To include specific files, start the line with the plus character '+'. To exclude files, start the line with the minus character '-'. You can use the following wildcards in the files names: *, **, ?.
9. Click **Save**.

What to do next

To add properties to the template, see Creating project template properties on page 386.

Creating build process templates

Build process templates provide properties, resources, and artifact sets to processes managed by the parent project.

Before you begin

Before you create a build process templates, create a job that you want to attach to the template. For information about creating jobs, see Creating jobs on page 376. If you want to configure resources for the template, create them before you start to work on the template. See, Creating lockable resources on page 330. If you want to configure artifact sets for the template, create them before you start to work on the template. See, Creating artifact sets on page 327.

1. On the left navigational panel, click **Templates > Process**.
2. Click **Create**.
3. Using the **New Process Template** dialog, type a name for the source template in the **Name** field.
4. Select **Build Process** from the **Type** list.
5. Select **Team** from the drop-down list.
6. Click **Save**.
   After you save the template, you can add properties, artifacts, and resources to it.
7. Define a process for the template. on page 390
8. Add a resource to the template.
   Resources can be added during design time, or added dynamically by a script. Resources are created on the **Lockable Resources** page. For more information, see, Creating lockable resources on page 330.

   a. Using the page for the process template, click **Add Resource**.

   b. Using the **Add Resource to Lock** dialog, determine whether you want to add the resource now, at design time, or add the resource at run time with a script.
      To add a resource at design time, select it from the **Explicit Resource** list. Otherwise, enter a BeanShell script in the **Scripted resource** area. The script must return the name of the resource, or a LockableResource object. To use a property, you use this syntax `return PropertyLookup.get("property_name")`. If the property returns null or an empty string, no lock is used during the build. If the name resource does not exist, it is created.

   c. To obtain an exclusive lock on the resource, select **Yes** for the **Exclusive** option.

   d. Click **Save**.

9. Optional. Add a property to the template.
   For information about creating properties, see Creating project template properties on page 386.
10. Optional. Configure artifact sets.
    Artifact sets are created with the **Artifact Sets** page. See, Creating artifact sets on page 327.

    a. Using the page for the process template, click **Artifacts**, then click **Add Artifact Config**.

    b. Using the **Artifact Configurations** pane, select an artifact set from the **Artifact Set** list.

    c. Specify the base directory in the **Base Directory** field.
       The directory is relative to the working directory when the Publish Artifacts step runs. To use the working directory, enter the period character '.'. You can use process and project properties to define the location, such as `${property: value}`.

    d. Specify the artifacts to retrieve in the **Include Artifacts** area.

You can use the * and ** wildcards to define the pattern. The ** wildcard indicates every directory; the* wildcard indicates every file. For example, the pattern `dist/**/*` retrieve every folder and file beneath the `dist`directory. You can also use process and project properties to define the pattern.

e. To exclude artifacts that would otherwise be included, specify them in the **Exclude Artifacts** area.
You can use process and project properties to define the pattern.

f. Click **Save**.

11. Click **Save**.

Creating process definitions

Process definitions define the jobs and steps in a build processes.

Before you begin

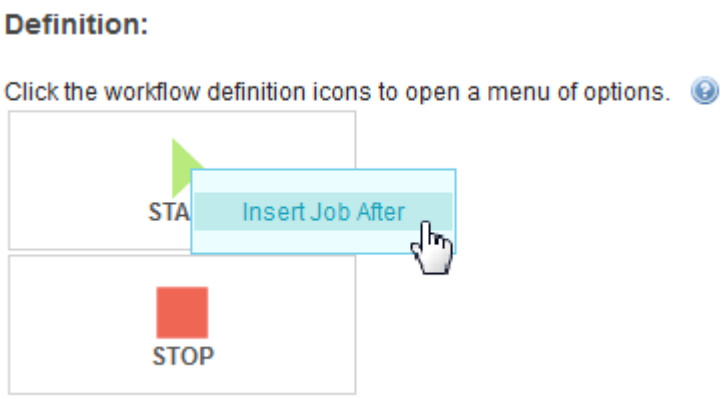
Process definitions are attached to process templates, so before you can define a process, you must create a process template.

About this task

You define processes by inserting jobs into the process and configuring properties that are related to the jobs. Steps are collected into jobs that are composed into processes. When you place a job in a process, you choose options such as the agent pool to run it, and the working directory, among other settings.

1. On the left navigational panel, click **Templates > Process**.
2. In the Process Templates page, select a process template, and then click **Definition**.
3. Using the **Definition** page's design area, define the process by completing these steps:

a. Click the **Start** icon in the **Definition** area, and select **Insert Job After**.
When you click a definition icon, a menu of related actions is displayed. If you click **Stop**, for example, the **Insert Job Before** action is available.



The following table describes all definition menu actions.

Table 20. Workflow action menu

Action	Description
Edit Configuration	Change the precondition script, agent filter, or working directory script on the job.
Edit Job Steps	Edit the individual job steps.
Insert Job After	Insert a job into the process. The selected job is the inserted job's parent and all of the selected job's children become the inserted job's children.
Insert Job Before	Insert a job into the process. The selected job is the child of the inserted job and the selected job's parents are the parents of the inserted job.
Add Parallel Job	Add a job to run in parallel to the selected job.
Iterate Job	Iterate the job a specified number of times.
Iteration Properties	Edit the properties specific to each job iteration.
Delete	Remove the job from the process definition.

4. Complete the following steps to define the properties that are related to the inserted job.

    a. Select the job that you want to insert into the process from the **Jobs** list.

    b. Select the job precondition from the **Pre-Condition** list.
    The job pre-condition script determines whether the job runs. For information about the job pre-condition scripts, see Creating step-precondition scripts on page 310. The precondition script runs before the agent filter runs.

    c. Use the **Agent Selection** options to specify an agent pool. To pick a specific pool, click **Agent Pool Selection** and select the pool from the list. The job runs on the first available agent in the selected pool. To select the pool by agent pool property, click **Agent Pool Property** and type the property, such as `${p:my.property}`. . If you do not want to use an agent or working directory for the job, select **No Agent**. This option requires that the job runs steps on the server, such as stamping or server-side scripts.

    d. Select a script from the **Working Directory** list.
    The selected script defines the path to the working directory for the job. The **Default Workflow Directory** script uses the agent's default directory, which is `{agent}/var/work/project/{project}/{workflow}`. The **Unique Job Directory** script uses a separate directory for each job: `{agent}/var/work/job/{job_id}`. The **Source Config's Working Directory** script uses the directory that is defined for the source configuration.

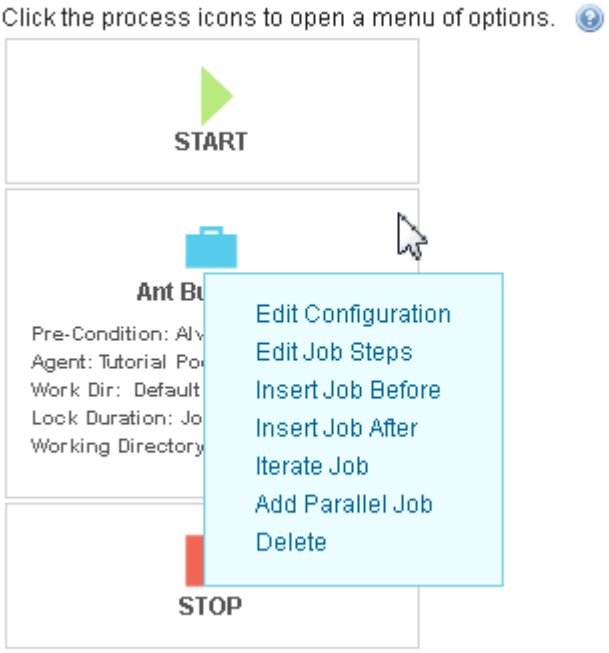
    e. Specify when you want the working directory cleaned by selecting an option from the **Working Directory Cleanup** list.

f. **Lock for Workflow**
Agents and their working directories are always locked when they process jobs. The **No** option locks the agent and working directory only during the current job. The **Yes** option locks the agent and working directory for the entire workflow and prevents other workflows from using the working directory.

5. Click **Save**.

Results

The job is inserted into the process, as shown in the following figure:



What to do next

Add more jobs to the process. Jobs can be inserted before or after other jobs. A process can have any number of jobs.

Creating secondary process templates

Secondary processes are processes that are run after a successful build, and build artifacts are generated and stored.

Before you begin

Before you create a secondary process template, create the project template to which you want to attach it. Create the workflow that you want to assign to the template. See, Creating process definitions on page 390. If you want to configure resources for the template, create them before you start to work on the template. See, Creating lockable resources on page 330.

About this task

Secondary processes inherit most of their configuration properties from the build workflow. After a secondary process is complete, the data is presented on the dashboard, under the same build life number as the parent build. Typically, secondary processes are used for testing.

1. On the left navigational panel, click **Templates > Process**.
2. Click **Create**.
3. Using the **New Process Template** dialog, type a name for the source template in the **Name** field.
4. Select **Secondary Process** from the **Type** list.
5. Select the **Team** from the drop-down list, and click **Save**.
6. Define a process for the template. on page 390
7. Add a resource to the template.
   Resources can be added during design time, or added dynamically by a script. Resources are created on the **Lockable Resources** page. For more information, see, Creating lockable resources on page 330.

   a. Using the page for the process template, click **Add Resource**.

   b. Using the **Add Resource to Lock** dialog, determine whether you want to add the resource now, at design time, or add the resource at run time with a script.
   To add a resource at design time, select it from the **Explicit Resource** list. Otherwise, enter a BeanShell script in the **Scripted resource** area. The script must return the name of the resource, or a LockableResource object. To use a property, you use this syntax `return PropertyLookup.get("property_name")`. If the property returns null or an empty string, no lock is used during the build. If the name resource does not exist, it is created.

   c. To obtain an exclusive lock on the resource, select **Yes** for the **Exclusive** option.

   d. Click **Save**.

8. Add properties to the template. See, Creating project template properties on page 386.
9. Click **Done**.

Monitoring and Reporting

With the monitoring tools you can review current and historical activity and save your filter conditions for future use. You can create charts and reports from most database tables.

Monitoring current activity

You can monitor current processes, agent activity, resource usage, builds, and schedules.

1. On the left navigational panel, click **Current Activity**.
   Result
   The Activity tab displays information about currently running processes, and processes queued to start.
2. If you want to filter the displayed information with a previously saved filter, select the filter from the **Saved Filters** area, then click **Filter**.
3. If you want to create a new search filter, complete the following steps.

a. **Optional:** Type a name for the filter in the **Current Filter** field.

b. **Optional:** Select an option from the **Group By** list, then use the set in current filter icon to add the grouping to the search filter.
Result
The group by option is added to the current filter. The **Context** option refers to related builds. A related build might be a build triggered by a dependency with another build. The **Project** option displays results by project. The **Stamp** displays results by build life stamp.

c. **Optional:** Select an option from the **Filter By** list.

d. **Optional:** Type the search criteria in the field below the **Filter By** list, and use the set in current filter icon associated with the **Filter By** list to add the criteria to the filter.
Result
The criteria is added to the current filter.

e. Click **Filter**.
Result
The results are displayed.

f. **Optional:** To save the filter for future use, click **Save**.
Result
The saved filter is displayed in the **Saved Filters** area.

4. **Optional:** To review current agent activity, click **Agents**.
Result
The **Agents** page displays activity for configured agents, and current directory locks. For information about agents, see Configuring agents on page 342.
5. **Optional:** To review the status of lockable resources, click **Lockable Resources**.
Result
The **Lockable Resources** page displays information about lockable resources and the projects that are using them. For information about lockable resources, see Creating lockable resources on page 330.
6. **Optional:** To review information about defined schedules, click **Schedules**.
Result
The **Schedules** page displays schedules and their next scheduled activities. For information about schedules, see Creating schedules on page 332.

What to do next

You can also monitor builds by using the **Information Radiator** dashboard. For information about the **Information Radiator** dashboard, see Information Radiator dashboard on page 395

Information Radiator dashboard

The **Information Radiator** dashboard provides real-time information about project builds.

The build processes are color-coded by build status, so that you can see at a glance the status of a build. When a build process runs, the status color changes to indicate the build's progress.

Each build process has a separate entry in the dashboard. If a project has several processes that are defined for it, each one has an entry. Build information is updated approximately every 5 seconds. The dashboard displays the following information for each build process:

- Project name and build process.
- Build time. The time is an average of previous builds. If the build process is running, the current build time is also displayed.
- A link to the **Main** tab for the build process.
- A link to the last or current build ID or stamp.
- A link that you can use to claim failed or aborted builds.
- The name of the failed build step for failed builds.
- A list of users who contributed source changes to a failed build.
- The name of the currently running build step for running builds.

The background color of a build entry indicates the build's final status. The dashboard uses the following color coding for build processes:

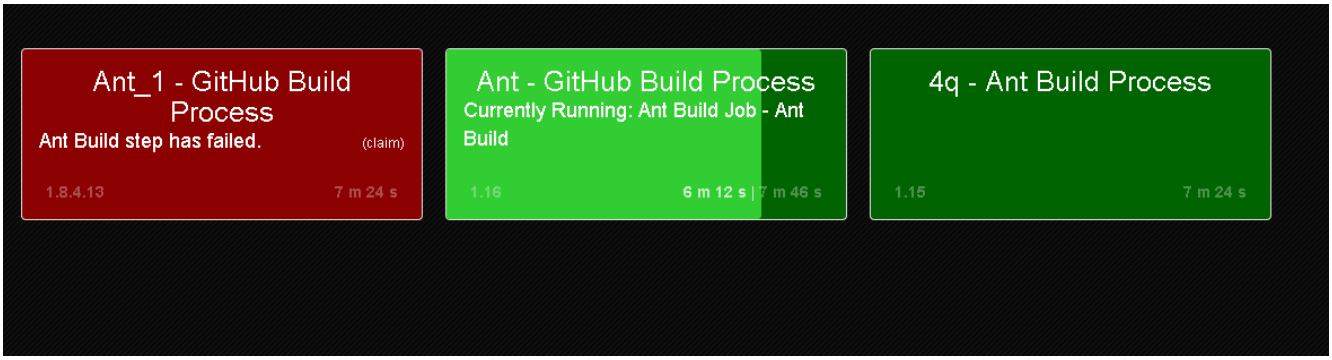
- Succeeded: dark green
- Failed: dark red
- Aborted: dark gray

You can customize the information that is displayed on the dashboard, and filter the information too. For information about customizing the **Information Radiator** dashboard, see Monitoring builds with the Information Radiator dashboard on page 396

Real-time build information

The background color of a build process changes when a build runs. The lighter-colored progress bar indicates the build's progress. Progress is calculated by comparing the current build time to the average time of previous builds. The name of the build's currently running job step is also displayed.

In the following figure, the Ant build process for the project that is named 4q succeeded, which is shown by the dark-green color of the entry, `4q - Ant Build Process`. You can see that the GitHub build process for the Ant project, `Ant - GitHub Build Process`, is running. The build's progress is shown by the lighter-colored progress bar. The name of the currently running step, `Ant Build Job - Ant Build`, is also displayed. The `Ant_1 - GitHub Build Process` build process failed, which is indicated by the dark-red color of the entry. The **claim** link is available for the failed build. If source changes were committed, the users who committed changes to the failed build are also listed. For information about claiming builds, see Monitoring builds with the Information Radiator dashboard on page 396.

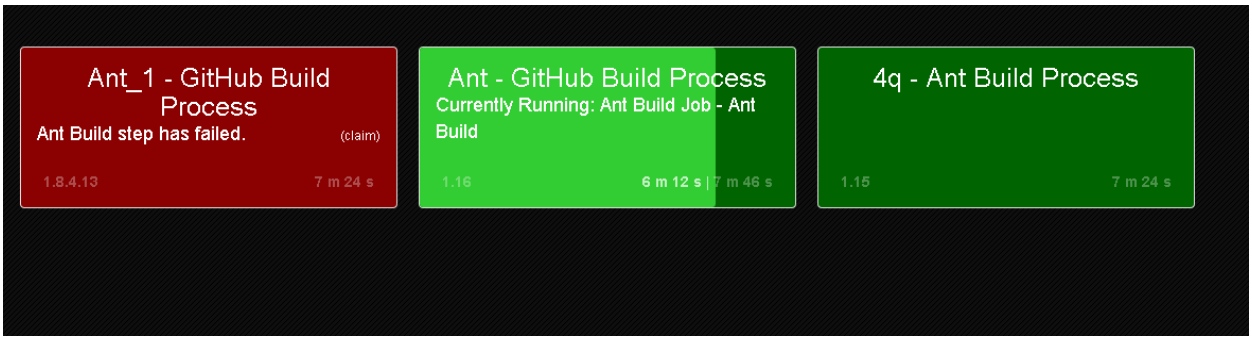


Monitoring builds with the Information Radiator dashboard

Use the **Information Radiator** dashboard to monitor builds.

About this task

You use the **Information Radiator** dashboard to monitor builds and claim failed builds. Anyone can access the dashboard. The build processes for which you have project permissions are available. Initially, each build process that ran at least once is displayed. You can filter the information and customize how the information is displayed. After customizing the dashboard, bookmark it to save your settings.

1. On the left navigational panel, click **Help >** **> Information Radiator**.
   Result
   The **Information Radiator** dashboard is displayed, as shown in the following figure.
   

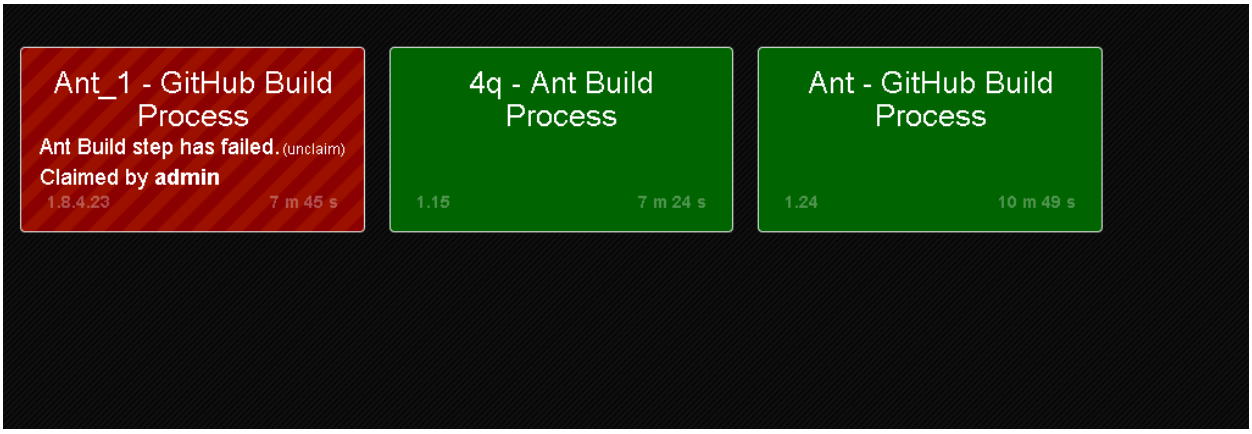
2. **Optional:** Click the name of a build process to display the **Main** tab for the process.
   The **Main** tab for a process provides historical information about the process, and you can use it to run the process.
3. **Optional:** Click the build ID or stamp to display the **Main** tab for the build life.
   The **Main** tab for a build life provides information about the build, and logs that you can use to troubleshoot the build.
4. **Optional:** To claim a failed or canceled build, complete these steps.
   a. Click **claim** for the failed or canceled build process.
   b. In the **Claim** dialog box, in the **User** field, enter the user ID of the user who is claiming the build process. By default, your user ID is entered.

c. In the **Reason** text box, enter your reason for claiming the build process.

d. Click **Done**.

Result

The background color of a claimed process becomes stripped when someone claims a build, as shown in the following figure. If a reason is entered, it is displayed as the entry in the dashboard. The user ID of the user who claimed the process is also displayed.



Note: Claim information is reset when the server is restarted.

5. **Optional:** Click **unclaim** to relinquish a claimed build.
   Any user can relinquish a build, not just the user who claimed it.
6. **Optional:** To customize the size of the build entries, complete thse steps.

   a. **Optional:** On the **Information Radiator** dashboard, click the **Settings** tool, .
   Result
   The **Settings** dialog box is displayed. The **Information Radiator** dashboard is divided into 12 columns. By default, all build entries are the same size, 4. The size refers to the number of entries per row. A setting of 4 means that each row displays 4 entries of 3 columns each (12/4=3). You can emphasize various build statuses by changing the size of build entries with that status.

   b. **Optional:** To change the size of build entries that have the failure status, in the **Failure Size Selector** list, select a column size.
   Build entriess with the failure status are adjusted to the selected column width.

   c. **Optional:** To change the size of entries for builds that are running, in the **Running Size Selector** list, select a column size.
   Entires for builds that are running are adjusted to the selected column width.

d. **Optional:** To change the size of build entries with the status of success, in the **Success Size Selector** list, select a column size.
Builds with the status of success are adjusted to the selected column width.

e. **Optional:** To change the size of builds with the status of aborted, in the **Aborted Size Selector** list, select a column size.
Entries for builds with the status of aborted are adjusted to the selected column width.

7. **Optional:** Click **Show Time Since Build** to display the amount of time that transpired since the last time the build ran.
8. **Optional:** To add or remove processes that are displayed on the dashboard, complete these steps.

a. Click **Add/Remove Processes**.
Result
The **Add/Remove Processes** dialog box is displayed. This dialog box displays all processes for which you have project permissions. By default, all processes are displayed.

b. Click **Select All** or **Select None** to display all processes.

c. Click a process or several processes to see only those processes.
Result
If some processes are selected, only those processes are displayed.

d. **Optional:** Click **Done**.

9. Click **Done**.
10. **Optional:** Bookmark the dashboard to save your settings.
Settings are saved in the URL. After you bookmark the dashboard, you can share your settings by sharing the URL.

Reporting

You can create reports about system activity, build lives, job execution, projects and their immediate dependencies, and several others. You can customize reports and save the results.

1. On the left navigational panel, click **Reporting**.
Result
The **Reporting** page provides tools to create and run reports.
2. Click **New Report**.
3. Type a name for the report in the **Name** field.
4. Select a table to add to the report from the **Tables** list, then click Select. **Select**.
Result
The table is added to the report and several options to customize the report are available.
5. **Optional:** Modify or add report columns by completing the following steps.

a. Click **Columns**.
Result

The columns for the selected table are displayed.

b. To rearrange column order, click a column and drag into the new location.

c. To rename a column, type the name in the **Alias** field that is associated with the column.

d. To remove a column from the report, use the delete icon associated with the column.

e. **Optional:** To add a function to a column, select the function from the **Function** list that is associated with the column.
The filter types are Boolean operators.

f. **Optional:** To add a compound column to the report, click **Add Compound Column** then type a name for the column in the **Included Columns** field.
All columns are included in the compound column. To remove a column, click the delete icon that is associated with the column.

g. Click **Apply**.

6. **Optional:** Filter the report by completing the following steps.

a. Click **Filter**.
Result
The columns for the selected table are available to be filtered.

b. Type a name for the report in the **Name** field.

c. Select a column to filter from the **Column** list.
The available columns depend on the selected table.

d. **Optional:** Select a type **Filter Type** list.
The filter types are Boolean operators.

e. **Optional:** Select a value from the **Value** options, then type or select a specific value.
Result
Selectable values depend on the table you choose.

f. Click **Apply**.

Result

The filter is added to the report. To delete a filter, click the remove icon . To modify a filter, click the edit icon .

7. **Optional:** Reorder report columns by completing the following steps.

a. Click **Order**.
Result
The columns for the selected table are available to be reordered.

b. Click **Add Ordering**.

c. Select the order key column from the **Column** list.
Result
The report is ordered by the selected column

d. **Optional:** Select an order direction from the **Order** list.
Columns can be in **Ascending** or **Descending** order.

e. **Optional:** Select a secondary order key from the **Column** list.

f. Click **Apply**.

8. **Optional:** Reorder report columns by completing the following steps.

a. Click **Order**.
Result
The columns for the selected table are available to be reordered.

b. Click **Add Ordering**.

c. Select the order key column from the **Column** list.
Result
The report is ordered by the selected column

d. **Optional:** Select an order direction from the **Order** list.
Columns can be in **Ascending** or **Descending** order.

e. **Optional:** Select a secondary order key from the **Column** list.

f. Click **Apply**.

9. **Optional:** If you want to represent the report's output in a chart, complete the following steps.

a. Click **Output**.
Result
Several chart types are available.

b. Define the chart's x-axis:
   i. Type the name for the x-axis in the **Name** field.
   ii. Select a column key from the **Value** list.
   iii. Select a column from the **Category** list. The default option is **None**.

c. Define the chart's y-axis:

i. Type the name for the y-axis in the **Name** field.
ii. Select a column key from the **Value** list.

d. Click **Apply**.

10. **Optional:** Modify the script that generates the report by completing the following steps.

a. Click **Text**.
Result
The JSON script that is generated by your selections is displayed. This script actually runs the report. You can directly modify the script.

b. Edit the JSON script.

11. Click **Run**.
Result
The report is displayed in the **Report** window. Report data is displayed in columns and rows unless you selected a chart-type output option.

What to do next

To save the generated report as a comma-separated values (CSV) file, use the download as CSV icon

.

Extending Product Functions

You can create your own plug-ins to extend the product's capabilities.

Plug-ins

Build plug-ins provide tools for creating workflows and integrating with external tools.

Plug-ins provide processing and integration functions. There are two types of plug-ins:

- *Source-type* plug-ins integrate with SCM systems.
- *Automation-type* plug-ins integrate with external systems and provide job steps.

Both types of plug-ins consist of distinct processes called *steps*. Each step consists of a number of properties, a command that runs the step, and post processing instructions. (Post processing typically ensures that expected results occur.) Step properties supply values to the plug-in step. Automation-type plug-in properties can be set at design time or at run time in the user interface. Source-type plug-in properties are typically defined when a component is created.

Automation-type plug-ins

You use automation plug-ins to run build job steps or otherwise manage builds. Each step is a self-contained unit with inputs, outputs, and properties. By combining automation plug-in steps, you can create fully automated build processes. When you create a job, you insert steps onto the design area and define the properties as you go. Property values can be specified when you define the process or at run time.

For information about creating jobs, see Creating jobs on page 376.

At build time, processes are run by agents that are installed in the target environment. For a process to run, the agent must have access to all resources, tools, and files that are required by the plug-in steps in the process. When you install an agent, ensure that these conditions are met:

- The agent that runs the build process has the necessary user permissions to run commands and can access required resources. Assigning these permissions typically entails granting permissions if an external tool is installed as a different user; installing the agent as a service; or impersonating the appropriate user.
- External tools that plug-in steps require are installed in the target environment.
- The required minimum version of an external tool is installed.

Source-type plug-ins

You use source plug-ins to import build artifacts and create builds. A source plug-in defines the type of artifacts that can be imported and identifies where the artifacts are located. Source-type plug-ins are configured when source items are created, you do not use the process editor to configure source-type plug-ins.

Source artifacts are imported by agents that are running in the environment where the artifacts are located. For source material to be created, the agent must have the required permissions and access to all required resources and files.

Viewing and obtaining plug-ins

A number of plug-ins come with the product and many others are available that can be downloaded and installed. For information about installing plug-ins, see Installing plug-ins on page 351.

View plug-ins that are installed by using the user interface. To view plug-ins, click **System > Plug-ins**. The plug-in type is displayed in the Type column.

Plug-ins can be developed and updated more frequently than Build provides release cycles.

Browse, download, and get documentation for plug-ins at the Build plug-ins website. The site contains many more plug-ins than are packaged with the Build product.

The site includes plug-ins that are developed by these sources:

IBM® Corporation

Your Build license entitles you to download and install the plug-ins that are developed by IBM® Corporation.

Support is provided through the developerWorks® community and through IBM® technical support.

Partners

Plug-ins that are provided by partners can be purchased from the partner website.

Support for each partner plug-in is provided exclusively by the partner who provided the plug-in.

Community

Plug-ins are provided by members of the Build community on Build plug-ins websiteBuild plug-ins website. Search with name of the plug-in to find plug-ins. You can contribute to code in the community or use code posted there to incorporate into your own.

Support is provided exclusively through the community.

For information about creating plug-ins, see Creating plug-ins on page 404.

Installing plug-ins

You install plug-ins by downloading the installation files and loading them into Build.

About this task

Plug-ins can be installed at any time. You do not have to restart the server after you install a plug-in. You can upgrade a plug-in by installing the new version over the old version.

1. Download the installation files for one or more plug-ins from Buildprovides plug-ins website.

   Documentation for plug-ins and the steps in the plug-ins is available on that site.

2. Load the plug-ins:
   a. On the left navigational panel, click **System > Plug-ins**.

   a. On the **Plug-ins** page, click **Browse** and select the downloaded plug-in file.
   You do not have to extract the compressed file.

   b. Click **Load**.

   c. Click **Done**.

Results

The plug-in is listed in **Loaded Plug-ins** list. After the plug-in is installed, it is available immediately.

Plug-ins at run time

Agents that are installed in the target environment run build processes. For a process to run, the agent must have access to all resources, tools, and files that are required by the plug-in steps that the process uses. When you install an agent, ensure that these conditions are met:

- The agent that runs the process has the required user permissions to run commands and access required resources. This permission status typically entails these actions:
  - Granting permissions if an external tool is installed as a different user
  - Installing the agent as a service
  - Impersonating the appropriate user

- External tools that plug-in steps require are installed in the target environment.
- The required minimum versions of external tools are installed.

Creating plug-ins

A plug-in consists of XML files and supporting script files that the plug-in requires.

- The `plugin.xml` file defines the steps that the plug-in consists of. A plug-in's steps define its functionality. Each step is an independently configurable entity in the Build editor.
- The `upgrade.xml` file is used to upgrade the plug-in to a new version.
- The `info.xml` file contains a version ID and other information that describes the plug-in.

Note: The plug-in files must use UTF-8 encoding.

A plug-in step is defined by a `<step-type>` element that contains these elements:

- One `<properties>` element.
- One `<command>` element.
- One `<post-processing>` element

The `<properties>` element is a container for `<property>` child elements, and can contain any number of `<property>` elements. You can specify property values design time or run time. The `<post-processing>` element provides error-handling capabilities and sets property values that can be used by other steps. The `<command>` element runs the step's function. The function can be defined completely by the element, or be constructed in part or entirely from the step's properties at design time or run time.

In addition to a step's own properties, a command has access to properties set earlier by other steps within the process.

Plug-in steps are run by an agent that is installed in the target environment. Thus, plug-ins can be written in any scripting language while the agent can access the required scripting tools on the host. After a plug-in is created, upload it into Build to make it available to users. To upload a plug-in, create an archive file that contains the XML files (`plugin.xml` and `upgrade.xml`) and scripts that the plug-in requires. Then, import the archive file with the **Plug-ins** page.

The `plugin.xml` file for automation plug-ins

An automation plug-in is defined with the `plugin.xml` file.

The structure of this file consists of a `header` element and one or more `step-type` elements. The `header` element identifies the plug-in. Each `step-type` element defines a step. Steps are available to users in the Build process editor and are used to construct component processes.

After the document-type declaration, the `plugin` root element identifies the XML schema type, `PluginXMLSchema_v1.xsd`, which all plug-ins use. The following code fragment presents the basic structure of the `plugin.xml` file for automation plug-ins:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://www.urbancode.com/PluginXMLSchema_v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <header>
    <identifier id="com.urbancode.air.plugin.Shell" version="4" name="Shell"/>
    <description>The Shell plugin allows users to run custom shell scripts during the deployment
 process.</description>
    <tag>Scripting/Shell</tag>
    <server:plugin-type>Automation</server:plugin-type>
  </header>
  <step-type name="Shell">
    <description>Execute a shell or batch script.</description>
    <properties>
      <property name="shellInterpreter">
        <property-ui type="textBox" label="Interpreter"
                        description="Optionally specify the interpreter to use to evaluate the command.
                        If left blank the following platform-specific behavior will be used:
                        Command Line will use .bat file interpreter on Windows and .com file interpreter on VMS.
 On other
                        systems it will use the shell specified by 'air/shell' agent variable if present. *nix
 systems
                        may override this by having the Command Line starts with a #! sequence."/>
      </property>
      <property name="scriptBody" required="true">
        <property-ui type="textAreaBox" label="Shell Script" description="Enter the script code to execute."/>
      </property>
      <property name="runAsDaemon">
        <property-ui type="checkBox" label="Daemon" description="Check this option to run the command in the
 background while permitting the step to
                complete immediately. The output will not be captured if running as a daemon."/>
      </property>
      <property name="outputFile">
        <property-ui type="textBox" label="Output File" description="Only set this value if Daemon is checked.
 Enter the
              path of the file to which the command output is directed. Leave blank to discard the output. The
 output
              will not be captured if an output file is specified."/>
      </property>
    </properties>
    <post-processing>
      <![CDATA[
        if (properties.get("exitCode") != 0) {
            properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
        }
        else {
            properties.put("Status", "Success");
        }
     ]]>
    </post-processing>

    <command program="${GROOVY_HOME}/bin/groovy">
      <arg value="-cp"/>
      <arg path="lib/shell.jar:lib/CommonsUtil.jar:lib/log4j.jar:lib/NativeProcess.jar:lib/WinAPI.jar"/>
      <arg file="shell.groovy"/>
      <arg file="${PLUGIN_INPUT_PROPS}"/>
      <arg file="${PLUGIN_OUTPUT_PROPS}"/>
    </command>
  </step-type>
```

For information about the structure of the `plugin.xml` file for source plug-ins, see Creating source plug-ins on page 407.

The `header` element

The mandatory `header` element identifies the plug-in and contains three mandatory child elements and one optional child element:

Table 21. Contents of the `header` element

`header` child elements	Description
`<identifier>`	This element's three attributes identify the plug-in: • `version` Specifies the API version (the version number for upgrading plug-ins is defined in the `info.xml` file). • `id` Identifies the plug-in. • `name` Defines the plug-in name that is shown on the **Automation Plugins** pane, and on informational pages for the plug-in. All values must be enclosed in single quotation marks.
`<description>`	Describes the plug-in. This element is shown on the **Automation Plugins** pane, and on informational pages for the plug-in.
`<tag>`	Defines where the plug-in is listed in the process editor's hierarchy of available plug-ins. The location is defined by a string that is separated by slashes. For example, the Tomcat definition is the following string: `Application Server/Java/Tomcat`. The Tomcat steps are listed after the Tomcat item, which in turn is nested under the other two. **Note:** Do not include underscore character (_) in this element.

The following sample shows a header element definition:

```
<header>
  <identifier version="3" id="com.&company;.air.plugin.Tomcat" name="Tomcat"/>
  <description>
  The Tomcat plugin is used during deployments to run Tomcat run-book
  automations and deploy or undeploy Tomcat applications.
  </description>
  <tag>Application Server/Java/Tomcat</tag>
</header>
```

Creating source plug-ins

Source-type plug-ins are defined with `upgrade.xml`, `info.xml`, and `plugin.xml` files.

Like automation plug-ins, source plug-ins are defined with three files: `upgrade.xml`, `info.xml`, and `plugin.xml`. The structure of the first two files is identical for both plug-in types. The structure of the `plugin.xml` file for the two plug-in types is different. Source-type plug-ins contain five SCM-type steps and the `server:plugin-type` property is `Source`.

The following code fragment contains a typical example of a source plug-in `header` element:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://www.urbancode.com/PluginXMLSchema_v1"
   xmlns:server="http://www.urbancode.com/PluginServerXMLSchema_v1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <header>
    <identifier id="com.urbancode.air.plugin.git" name="Git" version="4"/>
    <description>
       The Git Plugin automates cloning a Git repo, tagging source, and publishing source changes to the
 Changes tab of the Build Life.
    </description>
    <tag>SCM/Git</tag>
    <server:plugin-type>Source</server:plugin-type>
  </header>

    <!-- ==================================== -->
    <!--   SCM STEPS                          -->
    <!-- ==================================== -->

    <step-type name="Git Cleanup">
      <description>Perform a cleanup of the Git working directory</description>
      <properties>
        <property name="source">
          <server:property-ui type="sourcePropSheetRef"
                              label="Source Config"
                              hidden="true"/>
        </property>
      </properties>
      <post-processing><![CDATA[
        if (properties.get("exitCode") != 0) {
            properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
        }
        else {
            properties.put("Status", "Success");
        }
     ]]></post-processing>

      <command program="${GROOVY_HOME}/bin/groovy">
        <arg value="-cp"/>
        <arg path="classes"/>
        <arg file="cleanup.groovy"/>
        <arg file="${PLUGIN_INPUT_PROPS}"/>
        <arg file="${PLUGIN_OUTPUT_PROPS}"/>
      </command>

      <server:type>SCM_Cleanup</server:type>
      <server:validation language="javascript">
      </server:validation>
```

```
    </step-type>
```

Cleanup step

The Cleanup step requires the **sourcePropSheetRef** property. The step type must be `SCM_Cleanup`.

The following code fragment contains an example of a source plug-in `SCM_Cleanup` step:

```
<step-type name="Git Cleanup">
  <description>Perform a cleanup of the Git working directory</description>
  <properties>
    <property name="source">
      <server:property-ui type="sourcePropSheetRef"
                          label="Source Config"
                          hidden="true"/>
    </property>
  </properties>
  <post-processing><![CDATA[
    if (properties.get("exitCode") != 0) {
        properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
    }
    else {
        properties.put("Status", "Success");
    }
 ]]></post-processing>

  <command program="${GROOVY_HOME}/bin/groovy">
    <arg value="-cp"/>
    <arg path="classes"/>
    <arg file="cleanup.groovy"/>
    <arg file="${PLUGIN_INPUT_PROPS}"/>
    <arg file="${PLUGIN_OUTPUT_PROPS}"/>
  </command>

  <server:type>SCM_Cleanup</server:type>
  <server:validation language="javascript">
  </server:validation>
</step-type>
```

Checkout/Populate step

The Checkout/Populate step type must be `SCM_Populate`. The Checkout/Populate step has the following properties:

- **sourcePropSheetRef**
- **date**
- **cleanWorkspace**

The following code fragment contains an example of a source plug-in `SCM_Populate` step:

```
<step-type name="Git Checkout">
  <description>Perform a Git checkout of the workflow's source</description>
  <properties>
    <property name="cleanWorkspace">
```

```
      <server:property-ui type="checkBox"
                          label="Clean Workspace"
                          description="Erase all files from the workspace before performing checkout" />
    </property>
    <property name="date">
      <server:property-ui type="textBox"
                          label="Date"
                          description="Date of sourcecode to checkout ('yyyy-MM-dd HH:mm:ss z' OR
                          'E MMM dd HH:mm:ss zzz yyyy' OR milliseconds since Unix Epoch). Defaults to the
 current time." />
    </property>
    <property name="source">
      <server:property-ui type="sourcePropSheetRef"
                          label="Source Config"
                          hidden="true"/>
    </property>
  </properties>
  <post-processing><![CDATA[
    if (properties.get("exitCode") != 0) {
        properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
    }
    else {
        properties.put("Status", "Success");
    }
 ]]></post-processing>

  <command program="${GROOVY_HOME}/bin/groovy">
    <arg value="-cp"/>
    <arg path="classes"/>
    <arg file="checkout.groovy" />
    <arg file="${PLUGIN_INPUT_PROPS}"/>
    <arg file="${PLUGIN_OUTPUT_PROPS}"/>
  </command>

  <server:type>SCM_Populate</server:type>
  <server:validation language="javascript">
  </server:validation>
</step-type>
```

Label step

The Tag/Label step type must be `SCM_Label`. The Tag /Label step has the following properties:

- **sourcePropSheetRef**
- **message**
- **label**

The following code fragment contains an example of a source plug-in `SCM_Label` step:

```
<step-type name="Git Create Tag">
  <description>Create a tag in Git of the specified working copy.</description>
  <properties>
    <property name="message" required="true">
      <server:property-ui type="textBox"
                          label="Message"
```

```
                          description="A message for the commit" />
    </property>
    <property name="label" required="true">
      <server:property-ui type="textBox"
                          label="Tag"
                          description="The tag name to create" />
    </property>
    <property name="source">
      <server:property-ui type="sourcePropSheetRef"
                          label="Source Config"
                          hidden="true"/>
    </property>
  </properties>
  <post-processing><![CDATA[
    if (properties.get("exitCode") != 0) {
        properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
    }
    else {
        properties.put("Status", "Success");
    }
 ]]></post-processing>

  <command program="${GROOVY_HOME}/bin/groovy">
    <arg value="-cp"/>
    <arg path="classes"/>
    <arg file="tag.groovy"/>
    <arg file="${PLUGIN_INPUT_PROPS}"/>
    <arg file="${PLUGIN_OUTPUT_PROPS}"/>
  </command>

  <server:type>SCM_Label</server:type>
  <server:validation language="javascript">
  </server:validation>
</step-type>
```

Changelog step

The Changelog step type must be `SCM_Changelog`. The Changelog step has the following properties:

- **sourcePropSheetRef**
- **changesURL**
- **startBuildLifeId**
- **endDate**
- **startDate**

Use these properties to define the date range used when searching for changes.

The following code fragment contains an example of a source plug-in `SCM_Changelog` step:

```
<step-type name="Git Changelog">
  <description>Perform a Git changelog and publish the results.</description>
  <properties>
    <property name="startBuildlifeId">
      <server:property-ui type="textBox"
```

```
                        label="Start BuildLife Id"
                        description="The ID of the start buildlife. Captures changes since that build
 life." />
    </property>
    <property name="startDate">
      <server:property-ui type="textBox"
                        label="Start Date"
                        description="Start changelog at this date ('yyyy-MM-dd HH:mm:ss z' OR
                        'E MMM dd HH:mm:ss zzz yyyy' OR milliseconds since Unix Epoch)" />
    </property>
    <property name="endDate">
      <server:property-ui type="textBox"
                        label="End Date"
                        description="End changelog at this date ('yyyy-MM-dd HH:mm:ss z' OR
                        'E MMM dd HH:mm:ss zzz yyyy' OR milliseconds since Unix Epoch)" />
    </property>
    <property name="changesUrl">
      <server:property-ui type="textBox"
                        label="Changes URL"
                        hidden="true" />
    </property>
    <property name="source">
      <server:property-ui type="sourcePropSheetRef"
                        label="Source Config"
                        hidden="true"/>
    </property>
  </properties>
  <post-processing><![CDATA[
    if (properties.get("exitCode") != 0) {
        properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
    }
    else {
        properties.put("Status", "Success");
    }
 ]]></post-processing>

  <command program="${GROOVY_HOME}/bin/groovy">
    <arg value="-cp"/>
    <arg path="classes"/>
    <arg file="changelog.groovy"/>
    <arg file="${PLUGIN_INPUT_PROPS}"/>
    <arg file="${PLUGIN_OUTPUT_PROPS}"/>
  </command>

  <server:type>SCM_Changelog</server:type>
  <server:validation language="javascript">
  </server:validation>
</step-type>
```

QuietPeriod step

The QuietPeriod step type must be `SCM_QuietPeriod`. The QuietPeriod step has the following properties:

- **sourcePropSheetRef**
- **endDate**
- **startDate**

The QuietPeriod step typically runs the same command as the Changelog step. It always looks for the date of the latest change in the source repository.

The following code fragment contains an example of a source plug-in `SCM_QuietPeriod` step:

```
<step-type name="Git Quiet Period">
  <description>Perform a Git changelog for quiet period detection and publish most recent change date within
 the period.</description>
  <properties>
    <property name="startDate">
      <server:property-ui type="textBox"
                          label="Start Date"
                          description="Start changelog at this date ('yyyy-MM-dd HH:mm:ss z' OR
                          'E MMM dd HH:mm:ss zzz yyyy' OR milliseconds since Unix Epoch)" />
    </property>
    <property name="endDate">
      <server:property-ui type="textBox"
                          label="End Date"
                          description="End changelog at this date ('yyyy-MM-dd HH:mm:ss z' OR
                          'E MMM dd HH:mm:ss zzz yyyy' OR milliseconds since Unix Epoch)" />
    </property>
    <property name="source">
      <server:property-ui type="sourcePropSheetRef"
                          label="Source Config"
                          hidden="true"/>
    </property>
  </properties>
  <post-processing><![CDATA[
    if (properties.get("exitCode") != 0) {
        properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
    }
    else {
        properties.put("Status", "Success");
    }
 ]]></post-processing>

  <command program="${GROOVY_HOME}/bin/groovy">
    <arg value="-cp"/>
    <arg path="classes"/>
    <arg file="quietperiod.groovy"/>
    <arg file="${PLUGIN_INPUT_PROPS}"/>
    <arg file="${PLUGIN_OUTPUT_PROPS}"/>
  </command>

  <server:type>SCM_QuietPeriod</server:type>
  <server:validation language="javascript">
  </server:validation>
</step-type>
```

Plug-in steps: The `step-type` element

Plug-in steps are defined with the `step-type` element; each `step-type` represents a single step in the Build process editor.

A `step-type` element has a name attribute and several child elements:

- `description`
- `properties`
- `command`
- `post-processing`

The mandatory name attribute identifies the step. The description and name are shown in the web application and on informational pages for the plug-in..

```
<step-type name="Start">
<description>Start Apache HTTP server</description>
```

Note: A step name cannot contain the forward slash character (/).

Step properties: The `properties` element

The `properties` element is a container for properties, which are defined with the `property` tag.

Each step has a single `properties` element; a `properties` element can contain any number of `property` child elements.

A `property` tag has a mandatory `name` attribute, and an optional `required` attribute. This tag has two child elements: the `property-ui` element and `value` element, which are defined in the following table.

Table 22. The `property` element

`property` child elements	Description
`<server:property-ui>`	Defines how the property is presented to users in the Build editor. This element has several attributes: • `label` Identifies the property in the editor's **Edit Properties** dialog box. • `description` Contains the text that is displayed to users in the associated roll-over help box. • `default-value` Provides the property value that is displayed when the **Edit Properties** dialog box is displayed; used if unchanged.

Table 22. The `property` element (continued)

`property` child elements	Description
	• `type` Identifies the type of widget that is displayed to users. Possible values are as follows: ◦ `textBox` Enables users to enter an arbitrary amount of text, which is limited to 4064 characters. ◦ `textAreaBox` Enables users to enter an arbitrary amount of text in a larger input area than `textBox`. The limit is to 4064 characters. ◦ `secureBox` Accepts passwords. Similar to textBox except values are redacted. ◦ `checkBox` Displays a check box. If selected, a value of `true` is used; otherwise the property is not set. ◦ `selectBox` Requires a list of one or more values, which are displayed in a list. Configuring a value is described later.
`<value>`	Specifies values for a `selectBox` selection. Each value has a mandatory `label` attribute, which is displayed to users, and a value that is used by the property when selected. Values are displayed in the order they are defined.

The following code sample shows a `property` element definition:

```
<property name="onerror" required="true">
  <server:property-ui type="selectBox"
    default-value="abort"
    description="Action to perform when statement fails: continue, stop, abort."
    label="Error Handling"/>
  <value label="Abort">abort</value>
  <value label="Continue">continue</value>
  <value label="Stop">stop</value>
</property>
```

The `command` element

Steps are run by starting the scripting tool or interpreter specified by the `<command>` element. The `program` attribute defines the location of the tool that will perform the command.

It bears repeating that the tool must be located on the host and the agent invoking the tool must have access to it. In the following example, the location of the tool that will perform the command, the Java-based scripting tool *groovy* in this instance, is defined.

```
    <command program='${GROOVY_HOME}/bin/groovy'>
```

The actual command and any parameters it requires are passed to the tool by the `<arg>` child element. Any number of `<arg>` elements can be used. The `<arg>` element has several attributes:

Table 23. `<arg>` Attributes

Attribute	Description
`<value>`	Specifies a parameter passed to the tool. Format is tool-specific; must be enclosed by single-quotes or double-quotes.
`<path>`	Path to files or classes required by the tool. Must be enclosed by single-quotes or double-quotes.
`<file>`	Specifies the path to any files or classes required by the tool. Format is tool-specific; must be enclosed by single-quotes or double-quotes.

Becasue `<arg>` elements are processed in the order they are defined, ensured the order conforms to that expected by the tool.

```
    <command program='${GROOVY_HOME}/bin/groovy'>
      <arg value='-cp' />
      <arg path='classes:${sdkJar}:lib/commons-codec.jar:
                 lib/activation-1.1.1.jar:
                 lib/commons-logging.jar:lib/httpclient-cache.jar:
                 lib/httpclient.jar:lib/httpcore.jar:
                 lib/httpmime.jar:lib/javamail-1.4.1.jar' />
      <arg file='registerInstancesWithLB.groovy' />
      <arg file='${PLUGIN_INPUT_PROPS}' />
      <arg file='${PLUGIN_OUTPUT_PROPS}' />
   </command>
```

The `<arg file='${PLUGIN_INPUT_PROPS}' />` specifies the location of the tool-supplied properties file. The `<arg file='${PLUGIN_OUTPUT_PROPS}' />` specifies the location of the file that will contain the step-generated properties.

Note:

New lines are not supported by the `<arg>` element and are shown in this example only for presentation.

The `post-processing` element

When a plug-in step's `command` element finishes processing, the step's mandatory `post-processing` element runs.

The `post-processing` element sets the step's output properties and provides error handling. The `post-processing` element can contain any valid JavaScript™ script. Unlike the `command` element, `post-processing` scripts must be written in JavaScript™).

You have access to a *java.util.Properties* variable called `properties`. The `properties` variable has several special properties: `exitCode` contains the process exit code, and `Status` contains the step's status. A `Status` value of `Success` means that the step is complete. The Create File step's post-processing examines the command's `exitCode` and then sets the `Status` property accordingly.

Another available variable *scanner* can scan the step's output log and run commands that are based on the results.

You can use a post-processing script to set output properties that can be used in other steps in the same process, which enables complex workflows.

The script that is defined in the `post-processing` element is the step's default behavior. Users can also provide their own script, overriding the default behavior, when they define the step in the Build editor.

Upgrading plug-ins

The `upgrade.xml` file tells the server how to update a step or a property. The server scans every process that has been created on the server and upgrades the steps that you are upgrading.

The `upgrade.xml` file tells the server how to update a step or a property. The server scans every process that has been created on the server and upgrades the steps that you are upgrading.

To upgrade a plug-in, do the following:

1. Increment the number of the `version` attribute of the `<identifier>` element in `plugin.xml`.
2. Create a `<migrate>` element in `upgrade.xml` with a `to-version` attribute that contains the new number.
3. Place the property and step-type elements that match the updated `plugin.xml` file within the `<migrate>` element, as shown in the following example.

Considerations

Use the `<migrate-command>` element to upgrade steps.

- Upgrades to scripts and files for a step do not require changes to the `upgrade.xml` file. However, updating the version number in the `upgrade.xml` file is a good practice. For script changes to take affect, load the plug-in again.
- Keep a step by including a `<migrate-command>` element for each step to remain in the plug-in.

```
<migrate to-version="3">
    <migrate-command name="step_name"/>
</migrate>
```

- Add a new step does not require changes to the `upgrade.xml` file. Again, the version number of the plug-in should be updated.
- Delete a step by not including a `<migrate-command>` element for the steps to be removed from the plug-in.
- Rename a step by coding both the old and new name of the step using the `name` and `old` attributes.

```
<migrate to-version="3">
    <migrate-command name="new_step_name" old="old_step_name"/>
</migrate>
```

Use the `<migrate-properties>` elements to upgrade step properties.

- Delete a step property by using the `<remove-property>` element within the `<remove-properties>` element for the step with the removed property.

```
<migrate to-version="3">
    <migrate-command name="step_name"
        <remove-properties>
          <remove-property name="old_property_name" />
        </remove-properties>
   </migrate-command>
</migrate>
```

- Rename a step property by using the `name` and `old` attributes on the `<migrate-property>` element within the `<migrate-properties>` element for the step with the changed property.

```
<migrate to-version="3">
    <migrate-command name="step_name"
        <migrate-properties>
          <migrate-property name="new_property_name" old="old_property_name />
        </migrate-properties>
   </migrate-command>
</migrate>
```

- Change the default value for a step property by using the `default` attribute on the `<migrate-property>` element.

```
        <migrate-properties>
          <migrate-property name="new_property_name" old="old_property_name default="new_default_value" />
        </migrate-properties>
   </migrate-command>
</migrate>
```

Example

The following example shows how to add a `migrate` element in the upgrade.xml file to specify the command to use in the upgrade.

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin-upgrade
        xmlns="http://www.&company;.com/UpgradeXMLSchema_v1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <migrate to-version="3">
    <migrate-command name="Run SQLPlus script">
```

```
      <migrate-properties>
        <migrate-property name="sqlFiles" old="sqlFile"/>
      </migrate-properties>
      <remove-properties>
        <remove-property name="jvmProperties"/>
      </remove-properties>
    </migrate-command>
  </migrate>
  <migrate to-version="4">
    <migrate-command name="Run SQLPlus script" />
  </migrate>
  <migrate to-version="5">
    <migrate-command name="Run SQLPlus script" />
  </migrate>
</plugin-upgrade>
```

You can also make a script-only upgrade, which is an upgrade that contains changes to the step's associated scripts and files but does not change the `plugin.xml` file. This mechanism can be useful for plug-in development and for minor bug-fixes or updates.

The `info.xml` file

Use the `info.xml` file to describe the plug-in and provide release notes to users. The file's `<release-version>` element can be used for version releases.

This file is required for each plug-in.

Example plug-in

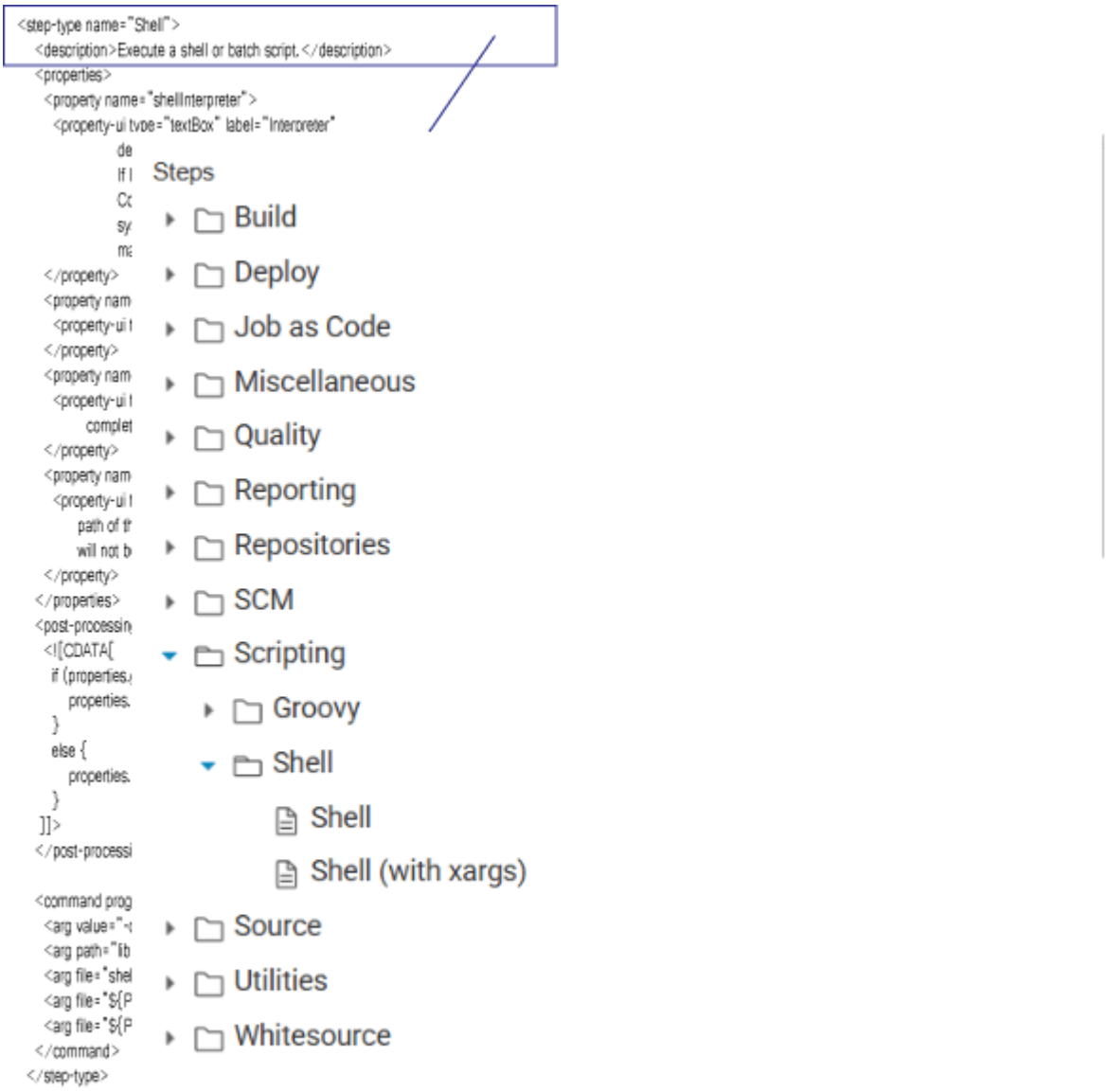
A plug-in consists of one or more steps. A step represents a unit of functionality that a user can configure and combine with other steps into a process. Creating a plug-in consists of defining its individual steps and then grouping them for presentation in Build.

Plug-in steps are run by an agent that is installed in the target environment. This means that plug-ins can be written in any scripting language provided that the agent can access the required scripting tools on the host.

The mechanics of plug-ins are covered by examining a plug-in step in detail. The example plug-in is *Shell*, which contains several steps that are related to shell scripts. The Shell plug-in comes with Build.

Each plug-in step is an individually configurable object. In the following illustration, you can see Shell plug-in's individual steps in the **Steps** window. The Shell step's label is derived from the `description` element in the plug-in file.

Plug-ins in the **Steps** window



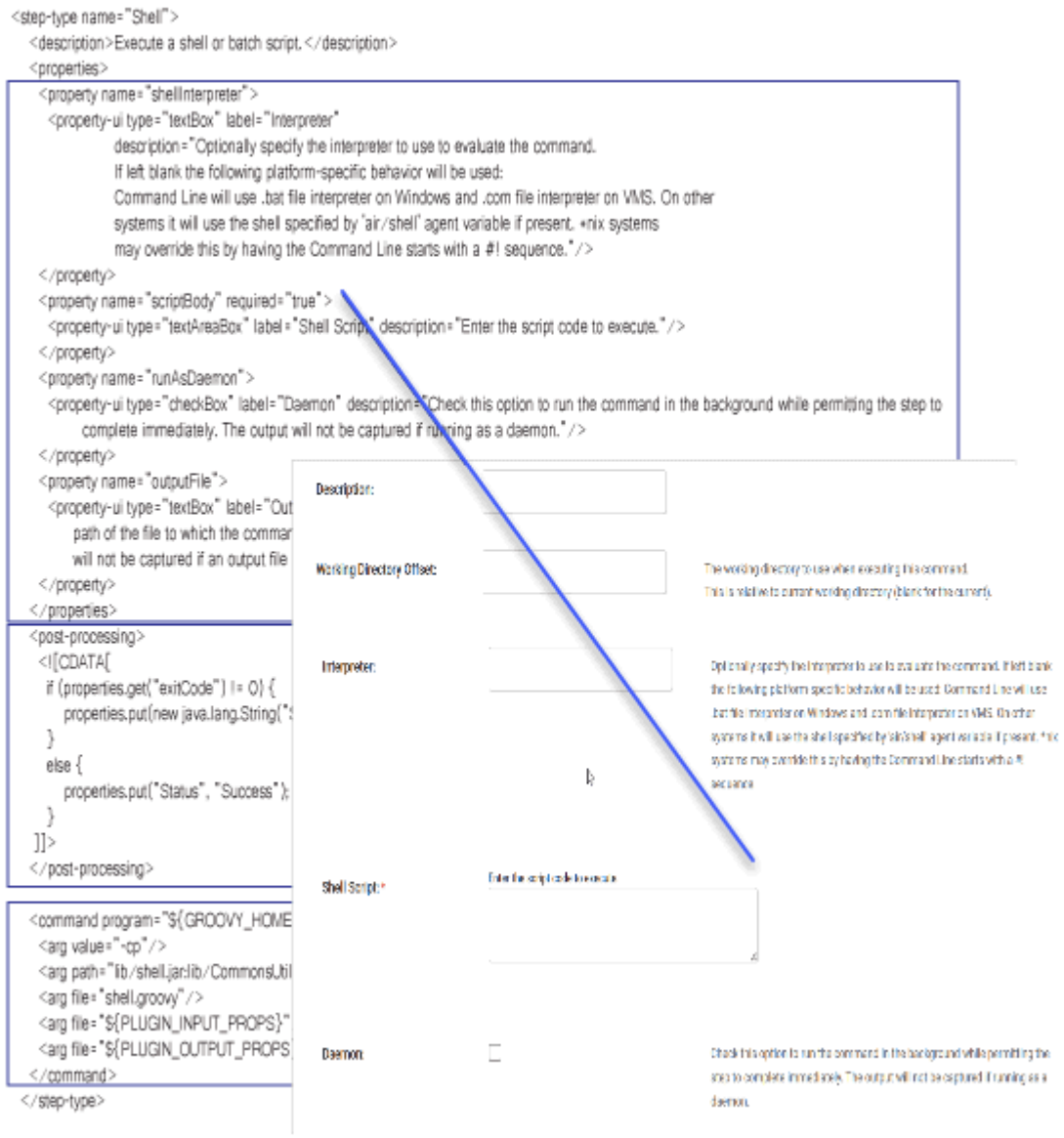
This example covers the *Shell* step. The Shell step executes a shell script or batch file that that you supply.

Each step (`step-type` element has the same structure:

- The `properties` element can contain any number of `property` child elements. Property values can be supplied at design time or run time.
- The `post-processing` element provides error-handling functions and sets property values that can be used by other steps.
- The `command` element performs the step's function. The function can be defined completely by the element, or be constructed in part or entirely from the step's properties at design time or run time.

The following figure shows how the Shell plug-in property definitions are displayed in the plug-in window.

Plug-in elements in the plug-in window



Step properties

In the context of plug-ins and steps, properties are values that the step's command uses. Step properties are defined with the `property` element.

The Create File step has three properties:

- The `file` property contains the name of the file that the command creates. This property is represented as a text box in the process designer
- The `contents` property contains the content of the file. This property is represented by a text-area box, which can contain a large amount of data
- The `overwrite` property specifies whether the file can overwrite an existing file. This property is represented by a check box in the process designer

These properties are displayed in the dialog box that opens when the step is added to a process. The other properties in the dialog are displayed for every step. Post processing is discussed later. Property values can be entered into the dialog box by the process designer at design time, or can be provided at run time by the user.

You can configure most properties with the `property-ui` child element (the `selectBox` type requires the `value` child element also). Default values can be defined when you create the step.

In addition to a step's own properties, a command has access to properties that are set earlier by other steps within the process, and to properties set by the process.

Step property values become unavailable after the process ends. Properties that are defined at run time are combined with the properties that were defined earlier and together are sent to the agent. Earlier properties (those properties that were defined outside the current process) are retrieved from the database. The way properties are processed and consumed is discussed further in the next section.

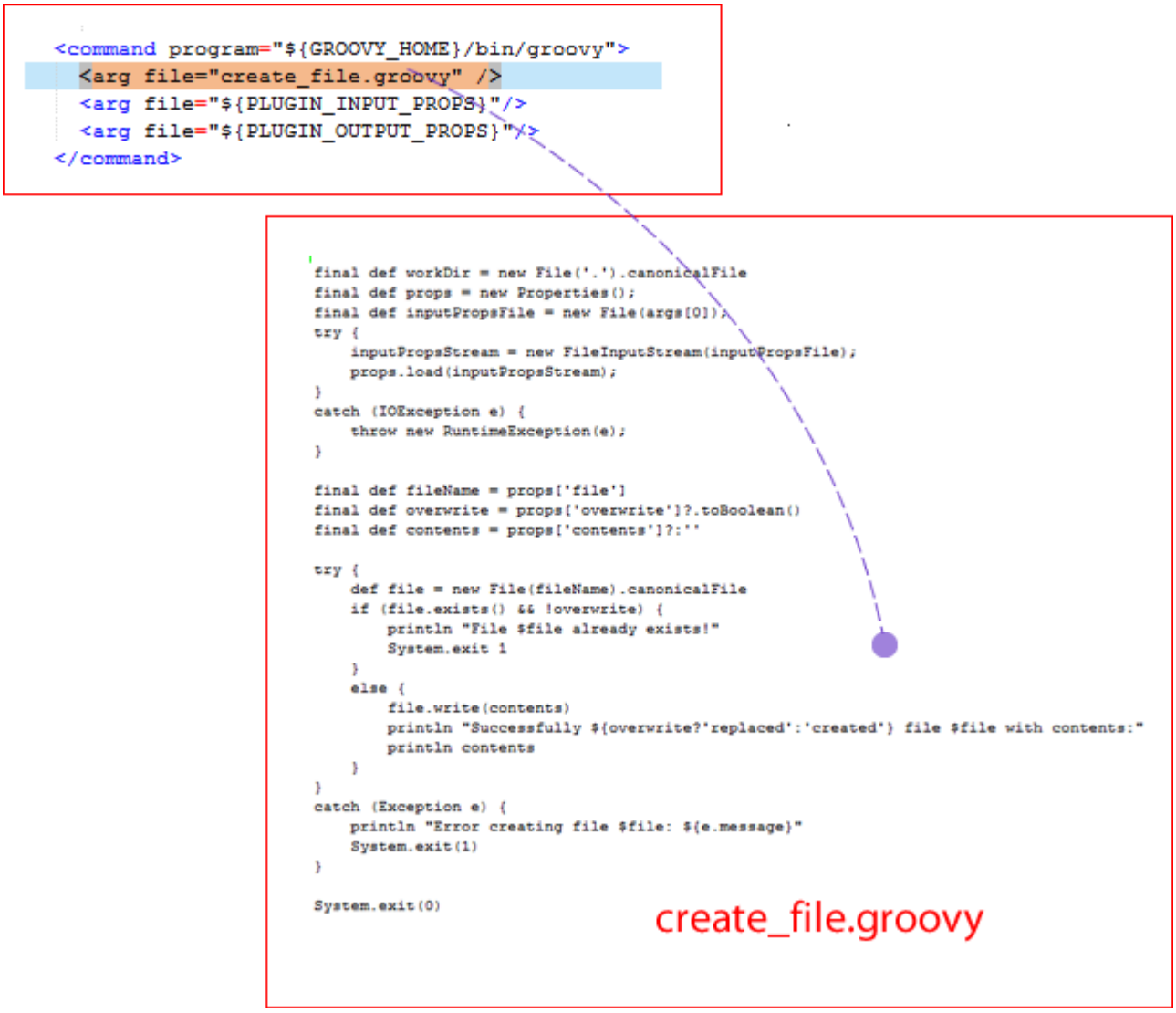
Step commands

The step's `command` element specifies the scripting tool that runs the step and identifies the file that contains the actual script.

The agent that runs the step downloads the plug-in and seeks the script among the downloaded files. The agent must also have access to the scripting tool. Other arguments that are required by the script or tool can also be specified with the `arg file` attribute. The `arg file` attributes must be defined so that they are compatible with the scripting tool.

The Create File command script is written in Groovy. Groovy is scripting language that is similar to Java™.

Figure 2. `create_file.groovy`



This command instructs the Groovy interpreter to run the `create_file.groovy` command. The next topic provides more information about Groovy and the particulars of this file.

The following line, which is part of every command, sends a file that contains the properties that are required by the step to the agent:

```
<arg file="${PLUGIN_INPUT_PROPS}"/>
```

The properties in the file are furnished at runtime and others that are defined earlier that are required by the step. The `${PLUGIN_INPUT_PROPS}` variable resolves to the location of this properties file.

The following line, which is also part of every command, refers to the file that the agent returns after it finishes the step:

```
<arg file="${PLUGIN_OUTPUT_PROPS}"/>
```

The properties in this file are available to later steps in the process. The `${PLUGIN_OUTPUT_PROPS}` variable resolves to the location of this properties file.

create_file.groovy

The `create_file.groovy` file contains the Groovy script that runs the step's command.

Groovy is a dynamic scripting language, which is similar to Python, Ruby, and Perl, for the Java™ platform. Most Java™ code is also syntactically valid Groovy, which makes Groovy popular with Java™ programmers. Groovy supports regular expressions.

The first lines of the script create a properties object, `props`. Then the script attempts to load the properties from the file that is sent from the server and which is specified by the `${PLUGIN_OUTPUT_PROPS}` variable. If it can load the file, it populates the `props` object; otherwise, it throws an exception.

```
final def workDir = new File('.').canonicalFile
final def props = new Properties();
final def inputPropsFile = new File(args[0]);
try {
inputPropsStream = new FileInputStream(inputPropsFile);
props.load(inputPropsStream);
}
catch (IOException e) {
throw new RuntimeException(e);
}
```

To run the command (create a file), the script uses the properties that are defined by the step itself. The script retrieves the three properties from the `props` object and creates corresponding local variables.

Next, the script creates a file with a name that is specified by `fileName` property, and tests the `overwrite` Boolean variable. If a file with the same name exists and `overwrite` is set to false, the script ends, or fails, with an exit code of 1. Exit codes can be examined during post-processing.

Otherwise, the file is written with the content of the `contents` property. A message is written to the output log, and the exit code is set to 0 (success).

```
final def fileName = props['file']
final def overwrite = props['overwrite']?.toBoolean()
final def contents = props['contents']?:''

try {
def file = new File(fileName).canonicalFile
if (file.exists() && !overwrite) {
println "File $file already exists!"
System.exit 1
}
else {
file.write(contents)
println "Successfully ${overwrite?'replaced':'created'} file
$file with contents:"
println contents
}
}
```

```
catch (Exception e) {
println "Error creating file $file: ${e.message}"
System.exit(1)
}

System.exit(0)
```

Build Scripting

The scripting API describes the various types of supported scripts.

The scripting API describes each type of script and how to create and maintain scripts for the Build server. Sample scripts are provided. Scripting objects and methods are described and examples are provided for each method.

To access the scripting API documentation, click the **Tools** link from any Build page.

Properties in Build

Properties can be set for many different things, including projects, build configurations, process configurations, build requests, and the server. You can also set and retrieve agent properties.

The scripting API can be found at the following location:

```
server_url/tools/scripting/ibm-ucb-scripting/
```

The API documentation contains many useful scripting examples.

Properties can be referenced by processes and passed to other objects. Property values can be defined when a property is created, or at some other point. The following syntax is used to reference properties:

```
${p:property_name}
```

You can also use this syntax:

```
${property:property_name}
```

If the property is not in the current context or scope, you add a prefix to the name to identify the context. The following syntax is used to reference properties outside the current context:

```
${p:prefix/property_name
```

For example, you can refer to a project property named `URL` by using the following syntax:

```
${p:project/URL}
```

To retrieve the value of a specific property, you can use the `@name=` syntax. For example, to retrieve the value of a project property that is named `specific.property`, you can use the following syntax:

```
${p:project/properties[@name='specific.property'}
```

If you do not use a prefix, the property is resolved by the property order of precedence. The following table provides the property order of precedence and the prefix that is used for every context. The property is always resolved in the current context, if possible; otherwise, the property is resolved according to the property order of precedence. The order is resolved from top of the list to bottom.

For example, if you have a variable that is named `URL` in both the step and job contexts, both contexts can refer to its own property by using the property name:

```
${p:URL}
```

If either context refers to the property in the other context, the prefix can be used. For example, the job can refer to the step property by using this syntax:

```
${p:step/URL}
```

If a step and a request both have a property that is named `URL`, then if a job refers to the property without using a prefix, the property resolves to the step property value. In this instance, if you want the job to refer to the request context, you must use the prefix: `${p:request/URL}`.

To retrieve the value of a property in another context that is itself retrieved from another context, you can chain prefixes. For example, to retrieve the project name from a build life, you might use this syntax:

```
${p:buildlife/workflow/project/name}
```

In most cases you can retrieve the project name with the more straightforward `${p:project/name}`, but this example illustrates how you might handle complicated contexts.

Table 24. Property order of precedence and context prefixes

Context	Description
Step	Context: Plug-in job steps. Syntax: `${p:step/property_name}`.
Job	Context: jobs. Syntax: `${p:job/property_name}`.
Request	Context: build requests. Syntax: `${p:request/property_name}`.
Source configuration (source plug-in steps only)	Context: source configuration template steps. Syntax: `${p:source/property_name}`.
Source configuration	Context: source configurations. Syntax: `${p:process/sourceConfigs/properties[@name='property_name']}`.

Table 24. Property order of precedence and context prefixes (continued)

Context	Description
	Source configurations are defined for build processes.
Process	Context: processes. Syntax: `${p:process/property_name}`.
Build life	Context: build life. Syntax: `${p:buildlife/property_name}`.
Build life	Context: build life. Syntax: `${p:buildlife/id}`. Returns the current build life ID. The Groovy equivalent is `${gvy:BuildLifeLookup.current.id}`
Build life build process	Context: build process. Syntax: `${p:originatingProcess/property_name}`.
Build life build request	Context: build request. Syntax: `${p:originatingRequest/property_name}`.
Project	Context: project. Syntax: `${p:project/property_name}`.
Server	Context: server. Syntax: `${p:property_name}`.

Table 24. Property order of precedence and context prefixes (continued)

Context	Description
	Note: There is no context prefix for server properties. Ensure that the names of your server properties are not duplicated by properties in other scopes.

Agent properties

Agent properties have a separate context. You can access agent properties directly, and you can access the agent environment variables. To refer to an agent property, use the following syntax:

```
${property_name}
```

You can also use this syntax:

```
${a:property_name}
```

or

```
${agent:property_name}
```

To access an agent environment variable, use the following syntax:

```
${env/PATH}
```

Property options

Many property elements are scripts. The scripts can be written in JavaScript™, Groovy, and BeanShell script. Scripts usually return an object or a value. Property prefixes are scripts that are used to dynamically reference properties in other contexts or scopes. The following table describes the property syntax options.

Table 25. Property syntax options

Property	Description
`${p:property_name}`	If defined as a non-agent property, return the property value. If the property is not defined, leave the property expression.
`${p?:property_name}`	If defined as a non-agent property, return the property value. If the property is not defined, return an empty string.
`${property_name}`	If defined as an agent property, return the property value. If the property is not defined, leave the property expression. You can also use the following syntax: `${a:property_name}`, or `${agent:property_name}`.

Table 25. Property syntax options (continued)

Property	Description
`${?:property_name}`	If defined as an agent property, return the property value. If the property is not defined, return an empty string. You can also use the following syntax: `${a?:property_name}`, or `${agent?:property_name}`.
`${gvy:script_body}`	Run `script_body` as Groovy and replace it with the return value.
`${bsh:script_body}`	Run `script_body` as BeanShell and replace it with the return value.

The `startsWith`, `contains`, and `matches` functions

You can retrieve all property values based on a property name. To retrieve all values, use the `startsWith`, `contains`, or `matches` functions, as shown in the following code examples:

```
    ${p:fn:startsWith(project/properties, 'prefix')}
    ${p:fn:contains(project/properties, 'text')}
    ${p:fn:matches(project/properties, 'prefix.*')}

    ${p:fn:startsWith(originatingProcess/properties, 'prefix')}
    ${p:fn:contains(originatingProcess/properties, 'text')}
    ${p:fn:matches(originatingProcess/properties, 'prefix.*')}

    ${p:fn:startsWith(process/properties, 'prefix')}
    ${p:fn:contains(process/properties, 'text')}
    ${p:fn:matches(process/properties, 'prefix.*')}

    ${p:fn:startsWith(request/properties, 'prefix')}
    ${p:fn:contains(request/properties, 'text')}
    ${p:fn:matches(request/properties, 'prefix.*')}

    ${p:fn:startsWith(buildlife/properties, 'prefix')}
    ${p:fn:contains(buildlife/properties, 'text')}
    ${p:fn:matches(buildlife/properties, 'prefix.*')}

    ${p:fn:startsWith(agent/properties, 'prefix')}
    ${p:fn:contains(agent/properties, 'text')}
    ${p:fn:matches(agent/properties, 'prefix.*')}
```

Cascading property values

Property values can be passed down from a parent process to a dependent process. The value can *cascade* from a direct dependent relationship or from an indirect relationship.

For example, using the table below Process A is the top level process that depends directly on Process B and Process C. Process B depends on Process D. Process C and Process D do not have any dependencies.

- Properties that meet the necessary requirements can be passed down from Process A to Process B and Process C.
- The property also cascades down to Process D, the inheritance flow is Process A to Process B to Process D.

Table 26. Dependent process example

Parent	Child	Child
Process A	Process B	Process C
Process B	Process D	

In order to cascade a property value to a dependent process, the following requirements must be met:

- ◦ The **Cascade Properties to Dependencies** option must be enabled. Click **System > Server Settings > Misc** to enable the option.
  ◦ Define the property on process template of the top level process.
    ▪ The property must be defined in both the parent process and the dependent processes.
    ▪ The **User May Override** option must be enabled for the property. This option is on the **Properties** page. This option indicates that there will be a prompt for the value of the property when the process runs instead of when the process is configured.

      Note: This option must be enabled on the process template for each dependent process that the property value is cascaded to if the dependent process is using a different process template than the high-level process.

  ◦ The dependency trigger between the high-level process and its dependent processes that the property is cascaded is either **Pull** or **Push**. The property value is not cascaded to a dependent process if the **No Trigger** is specified.

REST API reference

The Build server has a REST interface, which you can use to automate tasks on the server.

Running REST commands

Many different programs can run REST commands. To run the command, you call a method on a REST resource and pass parameters or a request in JSON format.

Note: Using REST commands requires the same permissions as using the web interface. For information about permissions, see Roles and permissions on page 319.

Example: Running simple REST commands with curl

The Linux™ program curl is a simple way to run REST commands. To run a REST command, put together the URL of one of the REST resources, specify the method to use, and add any parameters. For example, the following curl command retrieves a list of all initiatives. The command calls the GET method of the `agent` resource:

```
curl -k -u jsmith:passw0rd
  https://hostname:port/build/agents/
  -X GET -H "Accept: application/json"
```

Note: You must enter this command on a single line. It is split onto different lines here for clarity.

Note: This example uses the `-k` switch to connect to the server without checking SSL certificate validity. To set up authentication, see Authenticating for REST commands on page 432.

This example uses the user name `jsmith` and the password `passw0rd`. In most cases, create a dedicated user account for the REST commands to use and give that account the appropriate permissions.

Use the host name and port of your server for *`hostname`* and *`port`*. For example, if the host name is `ubuild.example.org` and the port is the default value of 8080, the `curl` command might look like the following example:

```
curl -k -u jsmith:passw0rd
  https://ubuild.example.org:8080/build/jobs/
  -X GET -H "Accept: application/json"
```

The response for this command is a JSONArray list of all jobs on the server, as in the following example:

```
[
  {
        "id": 32,
        "name": "Ant Build Job",
        "active": true,
        "description": null,
        "location": "/rest2/jobs/32"
    },
    {
        "id": 31,
        "name": "Demo Job",
        "active": true,
        "description": null,
        "location": "/rest2/jobs/31"
    },
]
```

Passing parameters to REST commands

Many REST commands have one or more parameters. Some parameters are added to the base of the URL and others are added at the end of the URL.

For example, the GET method of the `/jobs/{id}` resource returns information about a job. This method has a parameter in the base of its URL. The parameter **{id}** refers to the ID of a job. To use the GET method of this resource, you must substitute the ID of a job in the URL. For example, if the initiative ID is `31`, you might run the following command:

```
curl -k -u jsmith:passw0rd
  https://ubuild.example.org:8080/build/jobs/31/
  -X GET -H "Accept: application/json"
```

Some methods accept parameters at the end of the URL. For example, the GET method of the `projects` resource accepts the **format** parameter. This parameter specifies the format of the returned JSON. For example, to show additional detail about the projetcs on the server, pass the value `detail` to the **format** parameter, as in the following example:

```
curl -k -u jsmith:passw0rd
  https://ubuild.example.org:8080/build/projects/
  ?format=detail
  -X GET -H "Accept: application/json"
```

The resulting JSON code is more detailed than if you omitted the parameter.

Note: All parameter values must be URL encoded. For example, if a parameter includes a space, you must substitute the URL encoded value `%20`.

Passing JSON strings to commands

For more complex commands, you must send a JSON string or file instead of or in addition to the parameters.

For example, the POST method of the `initiatives` resource creates an initiative. To use this command, you must pass a JSON string that specifies the name and description of the new initiative. The JSON string for this command must follow this template:

```
{
  "name": "Initiative name",
  "description": "Initiative description"
}
```

This template is listed in the reference information for the command. See REST commands on page 439.

To pass this JSON string to the `projects` resource, you can either save the string to a file or include it in the command. For example, if you save the string to a file that is named `newProject.json`, the command looks like the following example:

```
curl -k -u jsmith:passw0rd
  https://ubuild.example.org:8080/build/projects/
  -d @newProjects.json
  -X POST -H "Content-Type: application/json"
```

You can also pass the string directly to the command, as shown in the following example:

```
curl -k -u jsmith:passw0rd
  https://ubuild.example.org:8080/build/projects/
  -d '{"name":"myProject","description":"My New Project"}'
  -X POST -H "Content-Type: application/json"
```

Note: The string that you pass to the command must be a valid JSON string.

Responses

After a successful command, most commands return either a simple string value or a JSON string.

In addition to the result of the command, the commands return standard HTTP status codes. The following list includes the most common status codes that REST commands return.

200

The command was successful.

400

- You do not have permission to run the command.
- You did not specify a required parameter.
- The resource path in the URL is incorrect.

404

The object that you are trying to retrieve does not exist.

405

The HTTP method name is incorrect.

415

The content type in the request header is incorrect.

500

The server encountered an error.

Authenticating for REST commands

The way that you authenticate to run REST commands depends on how the server is set up and the tool that you are using to run the commands.

Note: Using REST commands requires the same permissions as using the web interface. For information about permissions, see Roles and permissions on page 319.

Authenticating with a user name and password

The simplest way to authenticate for REST commands is to use a user name and password. For example, if you are using the `curl` program, you can specify the user name and password in the command, as in the following code:

```
curl -k -u jsmith:passw0rd
  https://ucrelease.example.org:8080/build/initiatives/
  -X GET -H "Accept: application/json"
```

Authenticating with a session key

For maximum performance, clients that perform multiple REST API requests should establish an HTTP session and reuse it for the duration of each batch of requests. An HTTP session is established after each successful request using HTTP Basic authentication and is identified by the `JSESSIONID` and `UCR_SESSION_KEY` cookies. In addition, clients that wish to use an HTTP session must place the `UCR_SESSION_KEY` into an HTTP request header for each request, as described below.

To get a session key, the client program first runs a command with a user name and password. The response headers include the session key. The header with the key typically has the name `Set-Cookie` and a value that is similar to the following example:

```
UCR_SESSION_KEY=6286eb44-b867-5243-875b-e61c5d8b4301;
Expires=Wed, 09-Sep-2082 16:51:45 GMT; Path=/; Secure
```

The client program must extract the session key from this header. In the previous example, the session key is `6286eb44-b867-5243-875b-e61c5d8b4301`. Then, for additional PUT, POST, and DELETE commands, include the following header:

```
UCR_SESSION_KEY:sessionKey
```

Use the session key for *`sessionKey`*.
In addition, to reuse the HTTP session, include the `JSESSIONID` cookie in the standard Cookie HTTP request header.

Authenticating in scripts and programs

Many programming and scripting languages can call REST commands.

The following example is a Python script that authenticates by adding the password to the request header.

```
#!/usr/bin/env python

import urllib2
import json
import base64
import sys

if not len(sys.argv) == 3:
  print 'usage: script <username> <password>'
  exit(1)

username = sys.argv[1]
password = sys.argv[2]

epass = base64.b64encode(username + ':' + password)
print 'base64 encoded: ' + epass
baseUrl = 'ubuild.example.org:8080'

url = 'http://' + baseUrl + '/applications/name'

opener = urllib2.build_opener(urllib2.HTTPHandler)
req = urllib2.Request(url)
req.add_header('Authorization', 'Basic '+epass)
req.get_method = lambda: 'GET'

resp = opener.open(req)
print resp.read()
```

Importing the server certificate

The default server certificate is unsigned. Some tools do not connect to servers with unsigned certificates by default. To access a server with a self-signed certificate, you can instruct the tool to connect insecurely, or you can import the certificate into your client. Follow these steps to import the certificate into your client:

1. Export the server certificate to a file:
    a. On the computer that hosts the Build server, open the `server.xml` file in a text editor. By default, this file is in the location *`server_install`*`/opt/tomcat/conf/server.xml`.
    b. In the `server.xml` file, find the following lines of code and note the values of the **keystoreFile** and **keystorePass** attributes:

    ```
    sslProtocol="TLS"
    keystoreFile="conf/tomcat.keystore"
    keystorePass="changeit" />
    ```

    c. In a command-line window, run the following command:

    ```
    keytool -v -list -keystore keyStoreFileName
    ```

    The `keytool` application is included in the Java™ developer kit and is not part of Build. Use the name of the **keystoreFile** attribute from the `server.xml` file for *`keyStoreFileName`*. When the command prompts you for a password, specify the value of the **keystorePass** attribute. The default value is `changeit`.
    d. From the result of the command, find the alias of the server. For example, the result of the command might look like the following code:

    ```
    Keystore type: JKS
    Keystore provider: SUN

    Your keystore contains 1 entry

    Alias name: server
    Creation date: Mar 2, 2016
    Entry type: PrivateKeyEntry
    ```

    In this code, the alias is `server`.
    e. Run the following command to export the certificate to a file and specify the password again:

    ```
    keytool -exportcert
      -alias serverAlias
      -keystore keyStoreFileName
      -storetype jks
      -file server.cert
    ```

    Use the alias of the server for *`serverAlias`*.
2. Copy the `server.cert` file to the client computer.
3. Import the `server.cert` file into the keystore of the client computer:
    a. In a command-line window on the client computer, run the following command and specify the password for the keystore on the client. The default is `changeit`.

    ```
    jreLocation\jre\bin\keytool.exe -importcert
      -alias serverAlias
      -file tomcat.cert
    ```

```
-storetype jks
-keystore jreLocation\jre\lib\security\cacerts
```

Use the location of the JRE or JDK for *`jreLocation`*.

Now, some tools that use this JRE or JDK accept the server certificate. Other tools, such as `curl`, might still not accept the server certificate because it is unsigned. To resolve this problem, set up a signed certificate for the server.

REST API conventions

The Build REST API follows a set of conventions to make interacting with it more consistent.

REST URLs

Each element type on the server is represented as a top-level URL with a plural form. In most cases, the URL uses the same terms as in the user interface, formatted with the camel case convention. For example, process templates are represented at the URL `http://base_url/build/processTemplates/`, agent pools are at the URL `http://base_url/build/agentPools/`, and workflows are at the URL `http://base_url/build/workflows/`, where `base_url` is the host name and path to the server.

Required HTTP headers

Most operations in the REST API accept an input in JSON format, return an output in JSON format, or both. Following HTTP conventions, the `Content-Type` request header is required for operations that provide JSON input, and the `Accept` request header is required for operations that produce JSON output, with the media type value of `application/json`. As a convenience for developers, the GET method for lists and individual items (described below) also produces JSON output if the media type is `text/html` and the query parameter `json` is appended to the request URL with any value. This parameter is useful for rendering JSON output from a web browser without special tools to modify the `Accept` header. For example, to display the JSON output for all plug-ins, type `http://base_url/build/plugins/?json` into the location bar of a web browser.

The `id` property

Elements such as projects have a unique ID (UUID) property, which appears in the JSON object representation as the `id` property as a JSON string. You can refer to a specific element by its UUID. You can use this UUID in places such as the URLs of REST API operations and within JSON representations of elements.

Basic operations

The basic create, read, update, delete operations are provided according to common REST API conventions. The top-level URL for an element type represents the collection of items of that type. To render the list of all elements of that type as an array of JSON objects, use the HTTP method GET with appropriate HTTP headers or the `json` query parameter. To create a new element, use the HTTP method POST to the top-level URL for the appropriate type and provide JSON data for the element in the body of the request. The response body includes a full JSON representation of the new element, including the server-generated `id` property. You can also provide an array of JSON objects to create multiple elements in a single request. In this case, the response body contains a JSON array of the new elements in the same order that they were provided in the request. Similarly, you can update the data for multiple elements by using the HTTP PUT method with a JSON array of elements. You can delete multiple elements by using the HTTP DELETE method with a JSON array that contains either full JSON elements or only the UUIDs of the elements.

To run an operation on a single existing element, append the UUID for the element to the top-level URL for the type. For example, to access the JSON data for the Ant plug-in that is provided, use the HTTP method GET with the URL `http://base_url/build/plugins/d43aa71e-5acc-4e28-8b4c-b19ee2b214be/`. To update the data for an element, use HTTP PUT method and provide an updated JSON object in the request body. (In this case, the UUID in the request URL takes precedence over the `id` property in the request body.) To delete the element, use the HTTP DELETE method; no request body is required in that case.

Output formats

When you use the create, read, and update operations (POST, GET, and PUT), you can include the optional **format** query parameter to adjust the JSON output for specific use cases. If you do not provide this parameter, or if the value of the parameter is not recognized, a default format is used. Each element type supports a different set of formats; these formats are listed with the reference information for the element type. Some formats show a large amount of detail and others show a small amount of detail, but the difference is only in the information that is shown; the values are the same.

You can use the `list` and `detail` formats with any element type. The `list` format provides summary information, such as you might show in a table of items. The `detail` format provides more detailed information. This format often includes the content of related elements. For some elements, these formats are identical. In addition, element types that have a `name` property also have a `name` format. This format includes only the `id` and `name` properties, which can be useful for displaying a list of items for selection by name. As a convenience, the `name` format for an element type can also be produced by appending `name` to the top-level URL for the type, for example, `GET http://base_url/build/jobs/name`.

JSON input conventions

When you provide JSON data as input for the create or update operations, the REST API takes into account only the properties that are writable on the element. The API ignores all other data. Read-only properties, such as computed counts or creation dates, are not updated. The API ignores unrecognized properties. Properties of related elements, other than the `id` property used to establish a reference, are also not updated.

For example, assume that you provide the following JSON input to create a project:

```
{
  "name": "New Project",
  "teams": "My Team",
  "totalChanges": 42,
  "tags": ["green"],
  "BadProperty": "xxxx"
}
```

When the API processes this input, it recognizes only the properties that are required for creating a release. Therefore, it ignores the unrecognized property `BadProperty` and the computed value `totalChanges`. In this case, the input above is equivalent to the simpler input below:

```
{
  "name": "New Project",
  "teams": "My Team",
  "tags": ["green"],
}
```

Similarly, when you update an existing element, the API changes only the properties that you specify in the JSON input. The API does not change properties that you omit. To clear the value of a property, specify an explicit JSON null value for that property.

For example, the following input changes the `tags` property in the sample project, but does not update any other properties:.

```
{
  "id": "123456789000",
  "name": "New Project",
  "tags": ["green","blue"],
  }
```

References

In most cases, you can refer to a single element with either a JSON string that contains the UUID of the element or a JSON object that contains an `id` property with the same value. As described above, any other properties of the referenced element are ignored. For example, all of the following JSON objects are equivalent when you are associating a change to a project.

```
{
  "id": "00000000-0000-0000-0000-123456789000",
  "project": "00000000-0000-0000-0000-000000000036"
}
```

```
{
  "id": "00000000-0000-0000-0000-123456789000",
  "project": {"id":"00000000-0000-0000-0000-000000000036"}
}
```

```
{
  "id": "00000000-0000-0000-0000-123456789000",
  "project": {
    "id":"00000000-0000-0000-0000-000000000036",
    "name":"Sample Project"}
}
```

```
{
  "id": "00000000-0000-0000-0000-123456789000",
  "project": {
    "id":"00000000-0000-0000-0000-000000000036",
    "name":"Not the actual Project name"}
}
```

In most cases, you can refer to a single element with either a JSON string that contains the UUID of the element or a JSON object that contains an `id` property with the same value. As described above, any other properties of the referenced element are ignored.

Multiple-value (collection) references

Some relationships can have multiple values. For example, a single project can be associated with multiple processes. In some cases, these relationships are not updated when you create or update the single-value side of the relationship. For example, when you create a project, the values that you provide for the `buildprocesses` property have no effect. Instead, use other URLs to add or remove build processes from projects.

Key-value properties

Some element types support the storage and retrieval of free-form key-value pairs in addition to statically defined properties. This type of property often represents data that is linked in some way.

Key-value properties are represented in the JSON data as a related element under the `properties` property. For example, the following code represents a change with key-value properties that are named `first`, `second`, and `third`:

```
{
  "id": "00000000-0000-0000-0000-123456789000",
  "name": "New Feature",
  "description": "",
  "status": "In Progress",
  "type": {
    "id":"00000000-0000-1201-0000-000000000001",
    "name":"Feature",
    "icon":"feature"},
  "properties": {
    "first":"first value",
    "second":"second value",
    "third":"third value"}
}
```

Both the key and value must be represented as JSON strings. As with statically defined properties, if a key is omitted from input, update operations do not change it. To clear a key-value pair, assign a JSON null value to the key.

Pagination

For some element types, the top-level GET request can provide a subset of elements rather than the full list. Retrieving subsets of elements like this is commonly referred to as "pagination," because it involves loading only a single "page" of data in each request.

Two alternative forms of paging are available, using either query parameters or the HTTP `Range` header.

To use query parameters for pagination, specify both a **rowsPerPage** and a **pageNumber** query parameter in the HTTP GET request. To retrieve the first page, specify the code `pageNumber=1`. For example, to retrieve the first five applications, use the request `GET http://base_url/build/jobs/?rowsPerPage=5&pageNumber=1`.

To use the HTTP `Range` header, specify the start and end values for the range of items, specifying zero-based and inclusive values. For example, to retrieve the first five jobs, add the following header to the request `GET http://base_url/jobs/`:

```
Range: items=0-4
```

Regardless of how you request the pagination, the HTTP response uses the `Content-Range` header to specify the actual range that is retrieved and the total number of available items. For example, if you requested the first five jobs of a list of 10, the response includes the following header:

```
Content-Range: 0-4/10
```

If you requested the first five job lives, and only three are available, the header looks like this:

```
Content-Range: 0-2/3
```

Sorting

To sort the results of the top-level GET operation, specify the **orderField** and **sortType** parameters. The **orderField** parameter must contain the name of a single property to sort by. The **sortType** parameter must contain either the value `asc` to sort in ascending order, or the value `desc` to sort in descending order.

REST commands

The following list shows the commands that are available.

agentPools resource

Get information about all agent pools

Return a JSON Array representation of all agent pools.

Request

```
GET https://{hostname}:{port}
  /build/rest2/agentPools
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "id": "The id of the agent pool.",
  "name": "The name of the agent pool."
}]
```

agents resource

Resources for working with agents.

Create a environment in agent environments

Request

```
POST https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/environment
```

Table 27. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Configure all unconfigured agents

Request

```
POST https://{hostname}:{port}
  /build/rest2/agents/configureAgents
```

Create a property in agent propertes

Request

```
POST https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/agent
```

Table 28. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Delete a specified name environment in agent environments

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/environment/{name}
```

Table 29. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent
name	string	true	The name of the environment

Delete a specified name property in agent propertes

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/agent/{name}
```

Table 30. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Table 30. URL parameters (continued)

Parameter	Type	Required	Description
name	string	true	The name of the property

Delete an agent

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/agents/{id}
```

Table 31. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get agent properties on an agent

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/agent?{parameters}
```

Table 32. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of agent properties to return per page.
pageNumber	string	false	The page of agent properties that should be returned.
orderField	string	false	The field to use when sorting the agent properties. This would be 'name'.
sortType	string	false	The order to use when sorting the agent properties. This would either be 'asc' or 'desc'.

Table 33. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get all properties on an agent

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}/properties
```

Table 34. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get environments on an agent

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/environment?{parameters}
```

Table 35. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of environment properties to return per page.
pageNumber	string	false	The page of environment properties that should be returned.
orderField	string	false	The field to use when sorting the environment properties. This would be 'name'.
sortType	string	false	The order to use when sorting the environment properties. This would either be 'asc' or 'desc'.

Table 36. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get information about all agents

Return a JSON representation of all agents.

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents?{parameters}
```

Table 37. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	If specified, the number of agents per page
pageNum	string	false	If specified, the page of agents to retrieve

Get information about an agent

Return a JSON representation of an agent.

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}?{parameters}
```

Table 38. Request parameters

Parameter	Type	Required	Description
extended	boolean	false	If set to true, retrieve information about the agent's properties.

Table 39. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get locked properties on an agent

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/locked?{parameters}
```

Table 40. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of locked properties to return per page.
pageNumber	string	false	The page of locked properties that should be returned.
orderField	string	false	The field to use when sorting the locked properties. This would be 'name'.
sortType	string	false	The order to use when sorting the locked properties. This would either be 'asc' or 'desc'.

Table 41. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Get system properties on an agent

Request

```
GET https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/system?{parameters}
```

Table 42. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of system properties to return per page.
pageNumber	string	false	The page of system properties that should be returned.
orderField	string	false	The field to use when sorting the system properties. This would be 'name'.
sortType	string	false	The order to use when sorting the system properties. This would either be 'asc' or 'desc'.

Table 43. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Update an agent

Request

```
PUT https://{hostname}:{port}
  /build/rest2/agents/{id}
```

Table 44. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agentPools": [{
    "dynamic": "Whether or not the agent pool is dynamic
  agent pool",
    "name": "The name of the agent pool"
  }],
  "description": "The agent description",
  "ignored": "Whether or not the server is ignoring the
  agent.",
  "maxJobs": "The Maximum number of jobs that can run on
  this Agent at any single time.",
  "name": "The name of the agent",
  "properties": [{
    "name": "Property name",
    "value": "Property value"
  }],
  "throughput": "The throughput metric setting is used to
  determine how likely an agent is to be used for another
  job.A weight is applied to the agent using a formula based
  on the throughput and the number of active jobs for the
  agent.A higher weight means that the agent is more likely
  to be chosen for an additional job.The formula is as
  follows: Weight = Throughput / (Number of Active Jobs + 1)"
}
```

Update environment in agent environments

Request

```
PUT https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/environment
```

Table 45. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

Update property in agent propertes

Request

```
PUT https://{hostname}:{port}
  /build/rest2/agents/{id}/properties/agent
```

Table 46. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an agent. This can be the name or ID of the agent

artifactsets resource

Create an artifact set

Request

```
POST https://{hostname}:{port}
  /build/rest2/artifactsets
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The description of the artifact set",
  "name": "The name of the artifact set"
}
```

Delete the artifact set

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/artifactsets/{id}
```

Table 47. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an artifact set. This is the ID of the artifact set

Get information about all artifact sets

Return a JSON representation of all artifact sets.

Request

```
GET https://{hostname}:{port}
  /build/rest2/artifactsets
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "The active status of the artifact set",
  "description": "The description of the artifact set",
  "id": "The id of the artifact set",
  "inactive": "The inactive status of the artifact set",
  "location": "The The URL that can be used to get the
  JSON representation of the artifact set",
  "name": "The name of the artifact set"
}]
```

Get information about the artifact set

Return a JSON representation of the artifact set

Request

```
GET https://{hostname}:{port}
  /build/rest2/artifactsets/{id}
```

Table 48. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an artifact set. This is the ID of the artifact set

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "The active status of the artifact set",
  "description": "The description of the artifact set",
  "id": "The id of the artifact set",
  "inactive": "The inactive status of the artifact set",
  "location": "The The URL that can be used to get the
  JSON representation of the artifact set",
  "name": "The name of the artifact set"
}
```

Update the artifact set

Request

```
PUT https://{hostname}:{port}
  /build/rest2/artifactsets/{id}
```

Table 49. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up an artifact set. This is the ID of the artifact set

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The description of the artifact set",
  "inactive": "The inactive status of the artifact set",
  "name": "The name of the artifact set"
}
```

authenticationRealm resource

authenticationRealm GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authenticationRealm
```

authenticationRealm POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/authenticationRealm
```

authenticationRealm/{realm} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}
```

Table 50. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}
```

Table 51. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}
```

Table 52. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm}/users GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}/users
```

Table 53. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm}/moveUp PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}/moveUp
```

Table 54. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm}/moveDown PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}/moveDown
```

Table 55. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm}/updateUsers PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}/updateUsers
```

Table 56. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authenticationRealm/{realm}/importUsers/{name} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authenticationRealm/{realm}/importUsers/{name}
```

Table 57. URL parameters

Parameter	Type	Required	Description
name	string	true	
realm	string	true	

authorizationRealm resource

authorizationRealm GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authorizationRealm
```

authorizationRealm POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/authorizationRealm
```

authorizationRealm/{realm} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authorizationRealm/{realm}
```

Table 58. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authorizationRealm/{realm} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authorizationRealm/{realm}
```

Table 59. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authorizationRealm/{realm} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/authorizationRealm/{realm}
```

Table 60. URL parameters

Parameter	Type	Required	Description
realm	string	true	

authtoken resource

authtoken GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authtoken
```

authtoken PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authtoken
```

authtokentable GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authtokentable
```

authtoken{token} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/authtoken{token}
```

Table 61. URL parameters

Parameter	Type	Required	Description
token	string	true	

authtoken{token} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/authtoken{token}
```

Table 62. URL parameters

Parameter	Type	Required	Description
token	string	true	

authtoken/getOrCreateToken PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/authtoken/getOrCreateToken
```

authtokens resource

Create an authtoken

Request

```
POST https://{hostname}:{port}
  /build/rest2/authtokens
```

This command takes a JSON request string or file. Use the following template for the request:

```
{}
```

Delete an authtoken

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/authtokens/{token}
```

Table 63. URL parameters

Parameter	Type	Required	Description
token	string	true	The authtoken string

Get information about queried tokens

Request

```
GET https://{hostname}:{port}
  /build/rest2/authtokens
```

Get information about the authtoken

Return a JSON representation of the authtoken

Request

```
GET https://{hostname}:{port}
  /build/rest2/authtokens/{token}
```

Table 64. URL parameters

Parameter	Type	Required	Description
token	string	true	The authtoken string

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "expiration": "The date and time at which this auth
  token should expire",
  "location": "The The URL that can be used to get the
  JSON representation of the anthtoken",
  "token": "The token string",
  "userId": "The security user id of the user who
  associates with this token",
  "userName": "The name of user who associates with this
  token"
}
```

build-lives resource

/build-lives/ GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives/?{parameters}
```

Table 65. Request parameters

Parameter	Type	Required	Description
includeInactive	string	false	

/build-lives//latest GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives/latest
```

/build-lives//latest/dependencies GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/dependencies
```

/build-lives//latest/dependencies POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//latest/dependencies
```

/build-lives//latest/artifact-configs GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifact-configs
```

/build-lives//latest/dependencies-xml GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/dependencies-xml
```

/build-lives//latest/artifactSets GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactSets
```

/build-lives//latest/artifactSets/{set} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactSets/{set}
```

Table 66. URL parameters

Parameter	Type	Required	Description
set	string	true	

/build-lives//latest/artifactSets/{set}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactSets/{set}/artifacts?{parameters}
```

Table 67. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 68. URL parameters

Parameter	Type	Required	Description
set	string	true	

/build-lives//latest/artifactSets/{set}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactSets/{set}/artifacts/{path}
```

Table 69. URL parameters

Parameter	Type	Required	Description
set	string	true	
path	string	true	

/build-lives//latest/artifactRepositories GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactRepositories
```

/build-lives//latest/artifactRepositories/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactRepositories/{id}
```

Table 70. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//latest/artifactRepositories/{id}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactRepositories/{id}/artifacts?{parameters}
```

Table 71. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 72. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//latest/artifactRepositories/{id}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/artifactRepositories/{id}/artifacts/{path}
```

Table 73. URL parameters

Parameter	Type	Required	Description
id	string	true	
path	string	true	

/build-lives//{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}
```

Table 74. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/dependencies GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/dependencies
```

Table 75. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/dependencies POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//{id}/dependencies
```

Table 76. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/artifact-configs GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifact-configs
```

Table 77. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/dependencies-xml GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/dependencies-xml
```

Table 78. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/artifactSets GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactSets
```

Table 79. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/artifactSets/{set} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactSets/{set}
```

Table 80. URL parameters

Parameter	Type	Required	Description
id	string	true	
set	string	true	

/build-lives//{id}/artifactSets/{set}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactSets/{set}/artifacts?{parameters}
```

Table 81. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 82. URL parameters

Parameter	Type	Required	Description
id	string	true	
set	string	true	

/build-lives//{id}/artifactSets/{set}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactSets/{set}/artifacts/{path}
```

Table 83. URL parameters

Parameter	Type	Required	Description
id	string	true	
set	string	true	
path	string	true	

/build-lives//{id}/artifactRepositories GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactRepositories
```

Table 84. URL parameters

Parameter	Type	Required	Description
id	string	true	

/build-lives//{id}/artifactRepositories/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactRepositories/{id}
```

Table 85. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/build-lives//{id}/artifactRepositories/{id}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactRepositories/{id}/artifacts?{parameters}
```

Table 86. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 87. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/build-lives//{id}/artifactRepositories/{id}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/artifactRepositories/{id}/artifacts/{path}
```

Table 88. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	
path	string	true	

Create a label.

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//latest/labels
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "label": "The name of the label.",
  "labelDependencies": "Determines which dependencies to
  which the label should also be applied. This is not
  required. The value should be 'NONE', 'DIRECT' or 'ALL'."
}
```

Create a label.

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//{id}/labels
```

Table 89. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "label": "The name of the label.",
  "labelDependencies": "Determines which dependencies to
  which the label should also be applied. This is not
  required. The value should be 'NONE', 'DIRECT' or 'ALL'."
}
```

Create a status.

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//latest/statuses
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "externalMessage": "The message of the status if the
  status is created by an external source.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "status": "The name of the status."
}
```

Create a status.

Request

```
POST https://{hostname}:{port}
  /build/rest2/build-lives//{id}/statuses
```

Table 90. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "externalMessage": "The message of the status if the
  status is created by an external source.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "status": "The name of the status."
```

```
}
```

Get a summary of all the findings in the source analytic report

Return a JSON representation of the summary of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analyticSummaries/{name}/summaries
```

Table 91. URL parameters

Parameter	Type	Required	Description
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get a summary of all the findings in the source analytic report

Return a JSON representation of the summary of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analyticSummaries/{name}/summaries
```

Table 92. URL parameters

Parameter	Type	Required	Description
id	string	true	
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
```

```
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get all the source analytic reports

Return a JSON representation of all source analytic reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analytics
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic report.",
    "count": "The count of findings in the source analytic
  report.",
    "id": "The id of the source analytic report.",
    "location": "The REST location to get the source
  analytic report.",
    "name": "The name of the source analytic report.",
    "type": "The type of the source analytic report.",
    "url": "The URL of the source analytic report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic reports

Return a JSON representation of all source analytic reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analytics
```

Table 93. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
```

```
  the source analytic report.",
    "count": "The count of findings in the source analytic
  report.",
    "id": "The id of the source analytic report.",
    "location": "The REST location to get the source
  analytic report.",
    "name": "The name of the source analytic report.",
    "type": "The type of the source analytic report.",
    "url": "The URL of the source analytic report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic summary reports

Return a JSON representation of all source analytic summary reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analyticSummaries
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
    "count": "The count of findings in the source analytic
  summary report.",
    "id": "The id of the source analytic summary report.",
    "location": "The REST location to get the source
  analytic summary report.",
    "name": "The name of the source analytic summary
  report.",
    "type": "The type of the source analytic summary
  report.",
    "url": "The URL of the source analytic summary report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic summary reports

Return a JSON representation of all source analytic summary reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analyticSummaries
```

Table 94. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
    "count": "The count of findings in the source analytic
  summary report.",
    "id": "The id of the source analytic summary report.",
    "location": "The REST location to get the source
  analytic summary report.",
    "name": "The name of the source analytic summary
  report.",
    "type": "The type of the source analytic summary
  report.",
    "url": "The URL of the source analytic summary report."
  }],
  "totalRecords": "The number of reports."
}
```

Get build lives for a project

Return the details of all the build lives for a project.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildLives?{parameters}
```

Table 95. URL parameters

Parameter	Type	Required	Description
id	string	true	Project id or project name

Table 96. Request parameters

Parameter	Type	Required	Description
limit	integer	false	Specifies the number of records to be fetched.
offset	integer	false	Indicates the starting point from which the data should be fetched.
extended	string	false	Boolean value to determine whether to fetch extended properties.

Table 96. Request parameters (continued)

Parameter	Type	Required	Description
modifiedSince	string	false	Represents the timestamp (in milliseconds) to fetch resources modified after this time.
preflightType	string	false	You can filter the build lives based on prefight type using the following values: • NOT_PREFLIGHT • ONLY_PREFLIGHT • ALL

Get information about all code coverage reports

Return a JSON representation of all code coverage reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/codeCoverage
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "branchPercentage": "The branch percentage of the code
  coverage report.",
    "buildLifeId": "The id of the build life that contains
  the code coverage report.",
    "groups": [{
      "branchPercentage": "The branch percentage of the
  group in the code coverage report.",
      "complexity": "The complexity of the group in the
  code coverage report.",
      "linePercentage": "The line percentage of the group
  in the code coverage report.",
      "methodPercentage": "The method percentage of the
  group in the code coverage report.",
      "name": "The name of the group in the code coverage
  report."
    }],
    "linePercentage": "The line percentage of the code
  coverage report.",
    "methodPercentage": "The method percentage of the code
  coverage report.",
    "name": "The name of the code coverage report.",
    "type": "The type of the code coverage report."
  }],
  "totalReports": "The number of the code coverage reports."
}
```

Get information about all code coverage reports

Return a JSON representation of all code coverage reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/codeCoverage
```

Table 97. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "branchPercentage": "The branch percentage of the code
  coverage report.",
    "buildLifeId": "The id of the build life that contains
  the code coverage report.",
    "groups": [{
      "branchPercentage": "The branch percentage of the
  group in the code coverage report.",
      "complexity": "The complexity of the group in the
  code coverage report.",
      "linePercentage": "The line percentage of the group
  in the code coverage report.",
      "methodPercentage": "The method percentage of the
  group in the code coverage report.",
      "name": "The name of the group in the code coverage
  report."
    }],
    "linePercentage": "The line percentage of the code
  coverage report.",
    "methodPercentage": "The method percentage of the code
  coverage report.",
    "name": "The name of the code coverage report.",
    "type": "The type of the code coverage report."
  }],
  "totalReports": "The number of the code coverage reports."
}
```

Get information about all issues.

Return a JSON representation of all issues.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/issues
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "count": "The number of the issues.",
  "issues": [{
    "id": "The id of the issue.",
    "name": "The name of the issue."
  }]
}
```

Get information about all issues.

Return a JSON representation of all issues.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/issues
```

Table 98. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "count": "The number of the issues.",
  "issues": [{
    "id": "The id of the issue.",
    "name": "The name of the issue."
  }]
}
```

Get information about all labels

Return a JSON Array representation of all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/labels
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "assignedDate": "The assigned date of the label.",
  "assignedDateFormat": "The format of the assigned date.",
  "buildLifeId": "The id of the build life.",
  "label": "The name of the label.",
  "labelId": "The id of the label.",
  "user": "The user that applied the label."
}]
```

Get information about all labels

Return a JSON Array representation of all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/labels
```

Table 99. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "assignedDate": "The assigned date of the label.",
  "assignedDateFormat": "The format of the assigned date.",
  "buildLifeId": "The id of the build life.",
  "label": "The name of the label.",
  "labelId": "The id of the label.",
  "user": "The user that applied the label."
}]
```

Get information about all of the findings in the source analytic report

Return a JSON representation of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analytics/{id}/findings
```

Table 100. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
```

```
  }],
  "totalRecords": "The number of findings."
}
```

Get information about all of the findings in the source analytic report

Return a JSON representation of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analytics/{id}/findings
```

Table 101. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get information about all source change sets

Return a JSON Array representation of all source change sets.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/sourceChange
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "comment": "The comment of the source change set.",
  "commitDate": "The commit date of the source change set.",
  "id": "The id of the source change set.",
  "paths": [{
    "path": "The path of the change.",
```

```
    "type": "The type of the change."
  }],
  "user": "The user name who submitted the source change
  set.",
  "version": "The version of the source change set."
}]
```

Get information about all source change sets

Return a JSON Array representation of all source change sets.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/sourceChange
```

Table 102. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "comment": "The comment of the source change set.",
  "commitDate": "The commit date of the source change set.",
  "id": "The id of the source change set.",
  "paths": [{
    "path": "The path of the change.",
    "type": "The type of the change."
  }],
  "user": "The user name who submitted the source change
  set.",
  "version": "The version of the source change set."
}]
```

Get information about all statuses

Return a JSON Array representation of all statuses.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/statuses
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "buildLifeId": "The id of the build life.",
  "dateAssigned": "The assigned date of the status.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "message": "The message of the status if the status is
  created by a user or external source.",
  "origin": "The origin of the status.",
```

```
  "status": "The status.",
  "statusColor": "The color of the status.",
  "statusName": "The name of the status.",
  "user": "The name of the user if the status is created
  by a user."
}]
```

Get information about all statuses

Return a JSON Array representation of all statuses.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/statuses
```

Table 103. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "buildLifeId": "The id of the build life.",
  "dateAssigned": "The assigned date of the status.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "message": "The message of the status if the status is
  created by a user or external source.",
  "origin": "The origin of the status.",
  "status": "The status.",
  "statusColor": "The color of the status.",
  "statusName": "The name of the status.",
  "user": "The name of the user if the status is created
  by a user."
}]
```

Get information about all test reports

Return a JSON representation of all test reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/tests
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "failure": "The number of failures in the test report.",
    "name": "The name of the test report.",
    "success": "The number of successes in the test
  report.",
```

```
    "successPercent": "The success percent of the test
  report.",
    "suites": [{
      "failure": "The number of failures in the test
  suite.",
      "name": "The name of the test suite.",
      "success": "The number of successes in the test
  suite.",
      "successPercent": "The success percent of the test
  suite."
    }]
  }],
  "successPercent": "The success percent of all the test
  reports."
}
```

Get information about all test reports

Return a JSON representation of all test reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/tests
```

Table 104. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "failure": "The number of failures in the test report.",
    "name": "The name of the test report.",
    "success": "The number of successes in the test
  report.",
    "successPercent": "The success percent of the test
  report.",
    "suites": [{
      "failure": "The number of failures in the test
  suite.",
      "name": "The name of the test suite.",
      "success": "The number of successes in the test
  suite.",
      "successPercent": "The success percent of the test
  suite."
    }]
  }],
  "successPercent": "The success percent of all the test
  reports."
}
```

Get information about the source analytic report

Return a JSON representation of the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analytics/{id}
```

Table 105. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic report.",
  "count": "The count of findings in the source analytic
  report.",
  "findings": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic report.",
  "location": "The REST location to get the source
  analytic report.",
  "name": "The name of the source analytic report.",
  "type": "The type of the source analytic report.",
  "url": "The URL of the source analytic report."
}
```

Get information about the source analytic report

Return a JSON representation of the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analytics/{id}
```

Table 106. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic report.",
  "count": "The count of findings in the source analytic
  report.",
  "findings": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic report.",
  "location": "The REST location to get the source
  analytic report.",
  "name": "The name of the source analytic report.",
  "type": "The type of the source analytic report.",
  "url": "The URL of the source analytic report."
}
```

Get information about the source analytic summary report

Return a JSON representation of the source analytic summary report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//latest/analyticSummaries/{name}
```

Table 107. URL parameters

Parameter	Type	Required	Description
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
```

```
  the source analytic summary report.",
  "count": "The count of findings in the source analytic
  summary report.",
  "findings": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic summary report.",
  "location": "The REST location to get the source
  analytic summary report.",
  "name": "The name of the source analytic summary report.",
  "type": "The type of the source analytic summary report.",
  "url": "The URL of the source analytic summary report."
}
```

Get information about the source analytic summary report

Return a JSON representation of the source analytic summary report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/build-lives//{id}/analyticSummaries/{name}
```

Table 108. URL parameters

Parameter	Type	Required	Description
id	string	true	
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
  "count": "The count of findings in the source analytic
  summary report.",
  "findings": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic summary report.",
  "location": "The REST location to get the source
  analytic summary report.",
  "name": "The name of the source analytic summary report.",
  "type": "The type of the source analytic summary report.",
  "url": "The URL of the source analytic summary report."
```

```
}
```

buildProcessTemplates resource

Resources for working with build process templates.

/buildProcessTemplates/{id}/workflow GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow
```

Table 109. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/buildProcessTemplates/{id}/workflow/graph GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph
```

Table 110. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/buildProcessTemplates/{id}/workflow/graph POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph
```

Table 111. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/buildProcessTemplates/{id}/workflow/graph PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph
```

Table 112. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/buildProcessTemplates/{id}/workflow/graph/insert POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/insert
```

Table 113. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/buildProcessTemplates/{id}/workflow/graph/{id} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}
```

Table 114. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/buildProcessTemplates/{id}/workflow/graph/{id}/iterate DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}/iterate
```

Table 115. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/buildProcessTemplates/{id}/workflow/graph/{id}/iterate POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}/iterate
```

Table 116. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 117. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 118. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName}
```

Table 119. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.
propertyName	string	true	The identifier of a job config.

Create a new property by copy on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/copyproperty
```

Table 120. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Create a process template

Request

```
POST https://{hostname}:{port}
  /build/rest2/buildProcessTemplates
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
```

```
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }]
}
```

Delete a job in the process definition.

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/workflow/graph/{id}
```

Table 121. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

Delete a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}
```

Table 122. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Delete a specified name property on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/property/{name}
```

Table 123. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
name	string	true	The name of the property

Delete the artifact on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/artifacts/{id}
```

Table 124. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	long	true	The id of the artifact

Delete the resource on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/resources/{seq}
```

Table 125. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
seq	int	true	The sequence of the resource

Get a list of processes that use the specified process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/processes?{parameters}
```

Table 126. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process templates to return per page.
pageNumber	string	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'asc' or 'desc'.

Table 127. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process using this process
  template is active",
  "description": "The description of the process using
  this process template",
  "id": "The ID of the process using this process template",
  "name": "The name of the process using this process
  template",
  "project": "The name of the project to which the process
  using this process template belongs"
}]
```

Get all artifacts on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/artifacts?{parameters}
```

Table 128. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template artifacts to return per page.
pageNumber	string	false	The page of process template artifacts that should be returned.

Table 129. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all properties on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/properties?{parameters}
```

Table 130. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template properties to return per page.
pageNumber	string	false	The page of process template properties that should be returned.

Table 131. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/resources
```

Table 132. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get avaiable resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/resourceOptions/{id}
```

Table 133. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	int	true	The id of the selected resource

Get information about a process template

Return a JSON representation of a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}
```

Table 134. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
```

```
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Get information about all build process templates

Return a JSON representation of all build process templates.

Request

```
GET https://{hostname}:{port}
  /build/rest2/buildProcessTemplates?{parameters}
```

Table 135. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of process templates to return per page.

Table 135. Request parameters (continued)

Parameter	Type	Required	Description
pageNumber	int	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'ASC' or 'DESC'.
filterValue_name	string	false	The name pattern of the process template to search for.
filterValue_type	string	false	The type of the process template to search for. Possible values are Any/Build/Secondary

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
```

```
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}]
```

Update a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}
```

Table 136. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
```

```
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Update artifacts on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/artifacts
```

Table 137. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update properties on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/properties
```

Table 138. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update resources on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/buildProcessTemplates/{id}/resources
```

Table 139. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

codestation-build-lives resource

/codestation-build-lives//latest GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest
```

/codestation-build-lives//{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}
```

Table 140. URL parameters

Parameter	Type	Required	Description
id	string	true	

Create a property in codestation buildLife

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest/properties
```

Create a property in codestation buildLife

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}/properties
```

Table 141. URL parameters

Parameter	Type	Required	Description
id	string	true	

Delete the Version of CodeStation project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest
```

Delete the Version of CodeStation project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}
```

Table 142. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get Versions of a CodeStation project

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives/?{parameters}
```

Table 143. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of Versions of a CodeStation project to return per page.
pageNumber	string	false	The page of Versions of a CodeStation project that should be returned.

Table 143. Request parameters (continued)

Parameter	Type	Required	Description
orderField	string	false	The field to use when sorting the Versions of a CodeStation project. This would be 'name' or 'description'.
sortType	string	false	The order to use when sorting the Versions of a CodeStation project. This would either be 'asc' or 'desc'.

Get the dependencies of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest/dependencies?{parameters}
```

Table 144. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of dependencies to return per page.
pageNumber	string	false	The page of dependencies that should be returned.
orderField	string	false	The field to use when sorting the dependencies. This would be 'codestationProjectName', 'startDate', 'id', 'status' or 'stamp'.
sortType	string	false	The order to use when sorting the dependencies. This would either be 'asc' or 'desc'.

Get the dependencies of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}/dependencies?{parameters}
```

Table 145. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of dependencies to return per page.

Table 145. Request parameters (continued)

Parameter	Type	Required	Description
pageNumber	string	false	The page of dependencies that should be returned.
orderField	string	false	The field to use when sorting the dependencies. This would be 'codestationProjectName', 'startDate', 'id', 'status' or 'stamp'.
sortType	string	false	The order to use when sorting the dependencies. This would either be 'asc' or 'desc'.

Table 146. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the properties of a Version of a CodeStation Project.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest/properties?{parameters}
```

Table 147. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of properties to return per page.
pageNumber	string	false	The page of properties that should be returned.
orderField	string	false	The field to use when sorting the properties. This would be 'name' or 'value'.
sortType	string	false	The order to use when sorting the properties. This would either be 'asc' or 'desc'.

Get the properties of a Version of a CodeStation Project.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}/properties?{parameters}
```

Table 148. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of properties to return per page.
pageNumber	string	false	The page of properties that should be returned.
orderField	string	false	The field to use when sorting the properties. This would be 'name' or 'value'.
sortType	string	false	The order to use when sorting the properties. This would either be 'asc' or 'desc'.

Table 149. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the statuses of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest/statuses?{parameters}
```

Table 150. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of statuses to return per page.
pageNumber	string	false	The page of statuses that should be returned.
orderField	string	false	The field to use when sorting the statuses. This would be 'name' or 'userName'.
sortType	string	false	The order to use when sorting the statuses. This would either be 'asc' or 'desc'.

Get the statuses of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}/statuses?{parameters}
```

Table 151. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of statuses to return per page.
pageNumber	string	false	The page of statuses that should be returned.
orderField	string	false	The field to use when sorting the statuses. This would be 'name' or 'userName'.
sortType	string	false	The order to use when sorting the statuses. This would either be 'asc' or 'desc'.

Table 152. URL parameters

Parameter	Type	Required	Description
id	string	true	

Update property in codestation buildLife

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-build-lives//latest/properties
```

Update property in codestation buildLife

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-build-lives//{id}/properties
```

Table 153. URL parameters

Parameter	Type	Required	Description
id	string	true	

codestation-projects resource

/codestation-projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}
```

Table 154. URL parameters

Parameter	Type	Required	Description
id	string	true	

/codestation-projects/{id}/build-lives/latest GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest
```

Table 155. URL parameters

Parameter	Type	Required	Description
id	string	true	

/codestation-projects/{id}/build-lives/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}
```

Table 156. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Create a codestation project

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-projects
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The codestation project description",
```

```
  "name": "The name of the codestation project",
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }]
}
```

Create a property in codestation buildLife

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest/properties
```

Table 157. URL parameters

Parameter	Type	Required	Description
id	string	true	

Create a property in codestation buildLife

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}/properties
```

Table 158. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Create a property in codestation project

Request

```
POST https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/properties
```

Table 159. URL parameters

Parameter	Type	Required	Description
id	string	true	

Delete the CodeStation project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}
```

Table 160. URL parameters

Parameter	Type	Required	Description
id	string	true	

Delete the Version of CodeStation project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest
```

Table 161. URL parameters

Parameter	Type	Required	Description
id	string	true	

Delete the Version of CodeStation project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}
```

Table 162. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Get CodeStation projects

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects?{parameters}
```

Table 163. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of CodeStation projects to return per page.
pageNumber	string	false	The page of CodeStation projects that should be returned.
orderField	string	false	The field to use when sorting the CodeStation projects. This would be 'name' or 'description'.
sortType	string	false	The order to use when sorting the CodeStation projects. This would either be 'asc' or 'desc'.

Get Versions for a new CodeStation project

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/new/build-lives
```

Get Versions of a CodeStation project

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives?{parameters}
```

Table 164. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of Versions of a CodeStation project to return per page.
pageNumber	string	false	The page of Versions of a CodeStation project that should be returned.
orderField	string	false	The field to use when sorting the Versions of a CodeStation project. This would be 'name' or 'description'.
sortType	string	false	The order to use when sorting the Versions of a CodeStation project. This would either be 'asc' or 'desc'.

Table 165. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get all properties on a CodeStation Project

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/properties?{parameters}
```

Table 166. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of CodeStation Project properties to return per page.
pageNumber	string	false	The page of CodeStation Project properties that should be returned.
orderField	string	false	The field to use when sorting the CodeStation Project properties. This would be 'name' or 'value'.
sortType	string	false	The order to use when sorting the CodeStation Project properties. This would either be 'asc' or 'desc'.

Table 167. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the dependencies of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest/dependencies?{parameters}
```

Table 168. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of dependencies to return per page.
pageNumber	string	false	The page of dependencies that should be returned.
orderField	string	false	The field to use when sorting the dependencies. This would be 'codestationProjectName', 'startDate', 'id', 'status' or 'stamp'.
sortType	string	false	The order to use when sorting the dependencies. This would either be 'asc' or 'desc'.

Table 169. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the dependencies of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}/dependencies?{parameters}
```

Table 170. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of dependencies to return per page.
pageNumber	string	false	The page of dependencies that should be returned.
orderField	string	false	The field to use when sorting the dependencies. This would be 'codestationProjectName', 'startDate', 'id', 'status' or 'stamp'.
sortType	string	false	The order to use when sorting the dependencies. This would either be 'asc' or 'desc'.

Table 171. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Get the properties of a Version of a CodeStation Project.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest/properties?{parameters}
```

Table 172. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of properties to return per page.
pageNumber	string	false	The page of properties that should be returned.
orderField	string	false	The field to use when sorting the properties. This would be 'name' or 'value'.
sortType	string	false	The order to use when sorting the properties. This would either be 'asc' or 'desc'.

Table 173. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the properties of a Version of a CodeStation Project.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}/properties?{parameters}
```

Table 174. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of properties to return per page.

Table 174. Request parameters (continued)

Parameter	Type	Required	Description
pageNumber	string	false	The page of properties that should be returned.
orderField	string	false	The field to use when sorting the properties. This would be 'name' or 'value'.
sortType	string	false	The order to use when sorting the properties. This would either be 'asc' or 'desc'.

Table 175. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Get the statuses of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest/statuses?{parameters}
```

Table 176. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of statuses to return per page.
pageNumber	string	false	The page of statuses that should be returned.
orderField	string	false	The field to use when sorting the statuses. This would be 'name' or 'userName'.
sortType	string	false	The order to use when sorting the statuses. This would either be 'asc' or 'desc'.

Table 177. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the statuses of a CodeStation Project Version.

Request

```
GET https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}/statuses?{parameters}
```

Table 178. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of statuses to return per page.
pageNumber	string	false	The page of statuses that should be returned.
orderField	string	false	The field to use when sorting the statuses. This would be 'name' or 'userName'.
sortType	string	false	The order to use when sorting the statuses. This would either be 'asc' or 'desc'.

Table 179. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Update a codestation project

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}
```

Table 180. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The codestation project description",
  "name": "The name of the codestation project"
}
```

Update property in codestation buildLife

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/latest/properties
```

Table 181. URL parameters

Parameter	Type	Required	Description
id	string	true	

Update property in codestation buildLife

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/build-lives/{id}/properties
```

Table 182. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

Update property in codestation project

Request

```
PUT https://{hostname}:{port}
  /build/rest2/codestation-projects/{id}/properties
```

Table 183. URL parameters

Parameter	Type	Required	Description
id	string	true	

conf/locale resource

/conf/locale/{language}.json GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/conf/locale/{language}.json
```

Table 184. URL parameters

Parameter	Type	Required	Description
language	string	true	

currentActivity resource

/currentActivity/agentDirLocks GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/currentActivity/agentDirLocks
```

/currentActivity/schedules GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/currentActivity/schedules
```

/currentActivity/lockableResources GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/currentActivity/lockableResources
```

dashboard resource

/dashboard/activity GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/dashboard/activity?{parameters}
```

Table 185. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of projects to return per page.
pageNumber	string	false	The page of projects that should be returned.

Table 185. Request parameters (continued)

Parameter	Type	Required	Description
sortType	string	false	The order to use when sorting the projects. This would either be 'asc' or 'desc'.
showInactive	boolean	false	When set to true, only show inactive projects. When set to false, only show active projects.

/dashboard/activity/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/dashboard/activity/{id}?{parameters}
```

Table 186. Request parameters

Parameter	Type	Required	Description
showInactive	boolean	false	When set to true, show all processes for inactive projects. When set to false, only show active workflos for active projects.

Table 187. URL parameters

Parameter	Type	Required	Description
id	long	true	

/dashboard/projects GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/dashboard/projects?{parameters}
```

Table 188. Request parameters

Parameter	Type	Required	Description
name	string	false	The project name pattern, * is allowed as a wildcard character
showInactive	boolean	false	When set to true, only show inactive projects. When set to false, only show active projects.

/dashboard/tags GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/dashboard/tags?{parameters}
```

Table 189. Request parameters

Parameter	Type	Required	Description
name	string	false	The tag name pattern, * is allowed as a wildcard character
showInactive	boolean	false	When set to true, only show inactive projects. When set to false, only show active projects.

dependencyConfig resource

Retrieve information about addable dependencies

Retrieve information about other processes that can bee added as dependencies to a process

Request

```
GET https://{hostname}:{port}
  /build/rest2/dependencyConfig?{parameters}
```

Table 190. Request parameters

Parameter	Type	Required	Description
fullName	string	false	The name pattern to use when looking up the dependency projects/processes to add.
projectId	long	true	The ID of the project to which dependencies are to be added.
processId	long	true	The ID of the process to which dependencies are to be added.

Example response sent by this endpoint:

```
{
  "fullName" : "The full name of the process in format \"Project - Process\" or the name of the codestation
 project",
  "processId" : "The ID of the process only if this is a non-Codestation project",
  "codestationId" : "The ID of the Codestation project only if this is a Codestation project"
}
```

dynamicAgentPools resource

Create a group rule

Request

```
PUT https://{hostname}:{port}
  /build/rest2/dynamicAgentPools/{id}
```

Table 191. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a dynamic agent pool.

This command takes a JSON request string or file. Use the following template for the request:

```
{"group": {
  "groups": "The list of group rules.",
  "logic": "The logic of the group rule. This should be
  'AND' or 'OR'.",
  "rules": [{
    "condition": "The condition of the rule. This should
  be 'IS_EQUAL_TO', 'IS_NOT_EQUAL_TO', 'CONTAINS',
  'DOES_NOT_CONTAIN', 'IS_GREATER_THAN', 'IS_LESS_THAN',
  'IS_GREATER_THAN_OR_EQUAL_TO', 'IS_LESS_THAN_OR_EQUAL_TO',
  'EXISTS' or 'DOES_NOT_EXIST'.",
    "conditionValue": "The value of the condition.",
    "type": "The type of the rule. This should be 'NAME',
  'PROPERTY', 'THROUGHPUT', 'MAX_JOBS' or 'DESCRIPTION'."
  }]
}}
```

Get information about all dynamic agent pools

Return a JSON Array representation of all dynamic agent pools.

Request

```
GET https://{hostname}:{port}
  /build/rest2/dynamicAgentPools
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "group": {
    "groups": "The list of group rules.",
    "logic": "The logic of the group rule. This should be
  'AND' or 'OR'.",
    "rules": [{
      "condition": "The condition of the rule. This should
  be 'IS_EQUAL_TO', 'IS_NOT_EQUAL_TO', 'CONTAINS',
  'DOES_NOT_CONTAIN', 'IS_GREATER_THAN', 'IS_LESS_THAN',
  'IS_GREATER_THAN_OR_EQUAL_TO', 'IS_LESS_THAN_OR_EQUAL_TO',
  'EXISTS' or 'DOES_NOT_EXIST'.",
      "conditionValue": "The value of the condition.",
      "type": "The type of the rule. This should be
  'NAME', 'PROPERTY', 'THROUGHPUT', 'MAX_JOBS' or
  'DESCRIPTION'."
```

```
    }]
  },
  "id": "The id of the dynamic agent pool.",
  "name": "The name of the dynamci agent pool."
}]
```

Get information about rule conditions

Return a JSON representation of rule conditions.

Request

```
GET https://{hostname}:{port}
  /build/rest2/dynamicAgentPools/ruleConditions
```

Get information about rule types

Return a JSON representation of rule types.

Request

```
GET https://{hostname}:{port}
  /build/rest2/dynamicAgentPools/ruleTypes
```

Get information about the dynamic agent pool

Return a JSON representation of the dynamic agent pool.

Request

```
GET https://{hostname}:{port}
  /build/rest2/dynamicAgentPools/{id}
```

Table 192. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a dynamic agent pool.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "group": {
    "groups": "The list of group rules.",
    "logic": "The logic of the group rule. This should be
  'AND' or 'OR'.",
    "rules": [{
      "condition": "The condition of the rule. This should
  be 'IS_EQUAL_TO', 'IS_NOT_EQUAL_TO', 'CONTAINS',
  'DOES_NOT_CONTAIN', 'IS_GREATER_THAN', 'IS_LESS_THAN',
  'IS_GREATER_THAN_OR_EQUAL_TO', 'IS_LESS_THAN_OR_EQUAL_TO',
  'EXISTS' or 'DOES_NOT_EXIST'.",
      "conditionValue": "The value of the condition.",
      "type": "The type of the rule. This should be
```

```
  'NAME', 'PROPERTY', 'THROUGHPUT', 'MAX_JOBS' or
  'DESCRIPTION'."
    }]
  },
  "id": "The id of the dynamic agent pool.",
  "name": "The name of the dynamci agent pool."
}
```

file resource

/file/{hash} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/file/{hash}
```

Table 193. URL parameters

Parameter	Type	Required	Description
hash	string	true	

files resource

/files POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/files
```

fixedAgentPools resource

Create a fixed agent pool

Request

```
POST https://{hostname}:{port}
  /build/rest2/fixedAgentPools
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agents": "The JSON Array of agent ids in the fixed
  agent pool.",
  "description": "The description of the fixed agent pool.",
  "name": "The name of the fixed agent pool.",
  "teams": "The JSON Array of team names for the fixed
  agent pool."
```

```
}
```

Delete the fixed agent pool.

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/fixedAgentPools/{id}
```

Table 194. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a fixed agent pool.

Get information about all fixed agent pool

Return a JSON Array representation of all fixed agent pool.

Request

```
GET https://{hostname}:{port}
  /build/rest2/fixedAgentPools
```

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "agents": "The JSON Array of agent ids in the fixed
  agent pool.",
  "description": "The dexcription of the fixed agent pool.",
  "id": "The id of the fixed agent pool.",
  "location": "The REST URI to get the information of
  fixed agent pool.",
  "name": "The name of the fixed agent pool."
}]
```

Get information about the fixed agent pool

Return a JSON representation of the fixed agent pool.

Request

```
GET https://{hostname}:{port}
  /build/rest2/fixedAgentPools/{id}
```

Table 195. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a fixed agent pool.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agents": "The JSON Array of agent ids in the fixed
  agent pool.",
  "description": "The dexcription of the fixed agent pool.",
  "id": "The id of the fixed agent pool.",
  "location": "The REST URI to get the information of
  fixed agent pool.",
  "name": "The name of the fixed agent pool."
}
```

Update the fixed agent pool

Request

```
PUT https://{hostname}:{port}
  /build/rest2/fixedAgentPools/{id}
```

Table 196. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a fixed agent pool.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agents": "The JSON Array of agent ids in the fixed
  agent pool.",
  "description": "The dexcription of the fixed agent pool.",
  "name": "The name of the fixed agent pool."
}
```

group resource

group GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/group
```

group POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/group
```

group/{group} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/group/{group}
```

Table 197. URL parameters

Parameter	Type	Required	Description
group	string	true	

group/{group} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/group/{group}
```

Table 198. URL parameters

Parameter	Type	Required	Description
group	string	true	

group/{group} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/group/{group}
```

Table 199. URL parameters

Parameter	Type	Required	Description
group	string	true	

group/{group}/members GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/group/{group}/members
```

Table 200. URL parameters

Parameter	Type	Required	Description
group	string	true	

group/{group}/members/{user} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/group/{group}/members/{user}
```

Table 201. URL parameters

Parameter	Type	Required	Description
user	string	true	
group	string	true	

group/{group}/members/{user} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/group/{group}/members/{user}
```

Table 202. URL parameters

Parameter	Type	Required	Description
user	string	true	
group	string	true	

import resource

/import PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/import
```

jobPreConditions resource

Get information about all job preConditions

Return a JSON representation of all job preConditions.

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobPreConditions
```

jobPreConditionScripts resource

/jobPreConditionScripts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobPreConditionScripts
```

/jobPreConditionScripts POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/jobPreConditionScripts
```

/jobPreConditionScripts/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobPreConditionScripts/{id}
```

Table 203. URL parameters

Parameter	Type	Required	Description
id	string	true	

/jobPreConditionScripts/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/jobPreConditionScripts/{id}
```

Table 204. URL parameters

Parameter	Type	Required	Description
id	string	true	

/jobPreConditionScripts/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/jobPreConditionScripts/{id}
```

Table 205. URL parameters

Parameter	Type	Required	Description
id	string	true	

jobs resource

/jobs/{id}/steps GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobs/{id}/steps
```

Table 206. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

/jobs/{id}/steps POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/jobs/{id}/steps
```

Table 207. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

/jobs/{id}/steps/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobs/{id}/steps/{id}
```

Table 208. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job
id	string	true	

/jobs/{id}/steps/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/jobs/{id}/steps/{id}
```

Table 209. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job
id	string	true	

/jobs/{id}/steps/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/jobs/{id}/steps/{id}
```

Table 210. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job
id	string	true	

Create a job

Request

```
POST https://{hostname}:{port}
  /build/rest2/jobs
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The job description",
  "name": "The name of the job"
}
```

Delete the job

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/jobs/{id}
```

Table 211. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

Get a JSON representation of process templates that use the specified job

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobs/{id}/processTemplates?{parameters}
```

Table 212. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process templates to return per page.
pageNumber	string	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'asc' or 'desc'.

Table 213. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

Get information about all jobs

Return a JSON representation of all jobs.

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobs?{parameters}
```

Table 214. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of jobs to return per page.
pageNumber	string	false	The page of jobs that should be returned.
orderField	string	false	The field to use when sorting the jobs. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the jobs. This would either be 'asc' or 'desc'.

Get information about the job

Return a JSON representation of the job

Request

```
GET https://{hostname}:{port}
  /build/rest2/jobs/{id}
```

Table 215. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "The ID of the project that contains the job",
  "description": "The job description",
  "id": "The ID of the job",
  "location": "The URL that can be used to get the JSON
  representation of the job",
  "name": "The name of the job"
}
```

Update the job

Request

```
PUT https://{hostname}:{port}
  /build/rest2/jobs/{id}
```

Table 216. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a job. This can be the name or ID of the job

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The job description",
  "name": "The name of the job"
}
```

label resource

Add a label to buildlives.

Request

```
POST https://{hostname}:{port}
  /build/rest2/label/addToBuild
```

Delete the label of buildlives.

Request

```
POST https://{hostname}:{port}
  /build/rest2/label/deleteLabel
```

Get all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/label/gridSelect?{parameters}
```

Table 217. Request parameters

Parameter	Type	Required	Description
name	string	false	

Get all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/label/itemFileReadStore?{parameters}
```

Table 218. Request parameters

Parameter	Type	Required	Description
name	string	false	

Replace the label of buildlives.

Request

```
POST https://{hostname}:{port}
  /build/rest2/label/replaceLabel
```

logs resource

/logsjob/{jobTraceId}/step/{stepSeq}/{fileName} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/logsjob/{jobTraceId}/step/{stepSeq}/{fileName}
```

Table 219. URL parameters

Parameter	Type	Required	Description
stepSeq	int	true	
jobTraceId	long	true	
fileName	string	true	

myactivity resource

/myactivity GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/myactivity
```

/myactivity/project/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/myactivity/project/{id}
```

Table 220. URL parameters

Parameter	Type	Required	Description
id	string	true	

/myactivity/workflow/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/myactivity/workflow/{id}
```

Table 221. URL parameters

Parameter	Type	Required	Description
id	string	true	

notificationSubscriptions resource

notificationSubscriptions GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/notificationSubscriptions
```

notificationSubscriptions POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/notificationSubscriptions
```

notificationSubscriptions/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/notificationSubscriptions/{id}
```

Table 222. URL parameters

Parameter	Type	Required	Description
id	string	true	

notificationSubscriptions/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/notificationSubscriptions/{id}
```

Table 223. URL parameters

Parameter	Type	Required	Description
id	string	true	

notificationSubscriptions/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/notificationSubscriptions/{id}
```

Table 224. URL parameters

Parameter	Type	Required	Description
id	string	true	

notificationSubscriptions/{id}/children GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/notificationSubscriptions/{id}/children
```

Table 225. URL parameters

Parameter	Type	Required	Description
id	string	true	

plugins resource

Create an integration

Request

```
POST https://{hostname}:{port}
  /build/rest2/plugins{id}/integrations
```

Table 226. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "propName": "propValue, if having multi properties,
  repeat this. If the propValue is an array, please use
  JSONArray format.",
```

```
  "propSheetName": "The name of the plugin"
}
```

Delete the integration

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/plugins{id}/integrations{id}
```

Table 227. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin
id	string	true	The identifier to use when looking up an integration. This is ID of the integration

Delete the plugin

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/plugins{id}
```

Table 228. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin

Get information about all integrations of the specified plugin

Return a JSON representation of all integrations of the specified plugin.

Request

```
GET https://{hostname}:{port}
  /build/rest2/plugins{id}/integrations
```

Table 229. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "propName": "propValue, if having multi properties,
  repeat this. If the propValue is an array, please use
  JSONArray format.",
  "propSheetId": "The id of the integration",
  "propSheetName": "The name of the plugin"
}]
```

Get information about all plugins

Return a JSON representation of all plugins.

Request

```
GET https://{hostname}:{port}
  /build/rest2/plugins?{parameters}
```

Table 230. Request parameters

Parameter	Type	Required	Description
type	string	false	The type of plugin. This would either be 'Automation' or 'Source'.

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "apiVersion": "The API version of the plugin",
  "commands": [{
    "description": "The description of the command",
    "id": "The id of the command",
    "name": "The name of the command",
    "postProcessingScript": "The post processing script of
  the command",
    "propSheetDefs": [{
      "description": "The description of the plugin",
      "id": "The id of the Property sheet",
      "name": "The name of the plugin",
      "properties": [{
        "allowedValuePropSheetDef": "The id of the
  PropSheetDef containing the allowed values for the
  property",
        "allowedValues": [{
          "id": "The id of the allowed value",
          "label": "The label of the allowed value",
          "value": "The value of the allowed value"
        }],
        "default-value": "The default value of the
  property",
        "description": "The description of the property",
        "hidden": "Whether or not the property is hidden",
        "id": "The id of the property",
        "label": "A label for this property shown when
  prompting users for a value",
```

```
      "name": "The name of the property",
      "required": "Whether or not the value of the
  property is required",
      "secure": "Whether or not the value of the
  property is secure",
      "type": "The type of the property",
      "value": "The value of the property. This is only
  used for properties on non-template objects (project, but
  not project template, etc)"
    }]
  }],
  "type": "The type of the command",
  "validationScript": "The validation script of the
  command",
  "validationScriptLanguage": "The language of the
  validation script"
  }],
  "description": "The description of the plugin",
  "hash": "The hash value of the plugin",
  "id": "The id of the plugin",
  "location": "The URL that can be used to get the JSON
  representation of the plugin",
  "name": "The name of the plugin",
  "pluginId": "The Plug-in ID of the plugin",
  "pluginVersion": "The version of the plugin",
  "propSheetDefs": [{
    "description": "The description of the plugin",
    "id": "The id of the Property sheet",
    "name": "The name of the plugin",
    "properties": [{
      "allowedValuePropSheetDef": "The id of the
  PropSheetDef containing the allowed values for the
  property",
      "allowedValues": [{
        "id": "The id of the allowed value",
        "label": "The label of the allowed value",
        "value": "The value of the allowed value"
      }],
      "default-value": "The default value of the property",
      "description": "The description of the property",
      "hidden": "Whether or not the property is hidden",
      "id": "The id of the property",
      "label": "A label for this property shown when
  prompting users for a value",
      "name": "The name of the property",
      "required": "Whether or not the value of the
  property is required",
      "secure": "Whether or not the value of the property
  is secure",
      "type": "The type of the property",
      "value": "The value of the property. This is only
  used for properties on non-template objects (project, but
  not project template, etc)"
    }]
  }],
  "tag": "The tag of the plugin",
  "type": "The type of the plugin"
}]
```

Get information about the integration

Return a JSON representation of the integration

Request

```
GET https://{hostname}:{port}
  /build/rest2/plugins{id}/integrations{id}
```

Table 231. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin
id	string	true	The identifier to use when looking up an integration. This is ID of the integration

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "propName": "propValue, if having multi properties,
  repeat this. If the propValue is an array, please use
  JSONArray format.",
  "propSheetId": "The id of the integration",
  "propSheetName": "The name of the plugin"
}
```

Get information about the plugin

Return a JSON representation of the plugin

Request

```
GET https://{hostname}:{port}
  /build/rest2/plugins{id}
```

Table 232. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "apiVersion": "The API version of the plugin",
  "commands": [{
    "description": "The description of the command",
    "id": "The id of the command",
    "name": "The name of the command",
    "postProcessingScript": "The post processing script of
```

```
the command",
  "propSheetDefs": [{
    "description": "The description of the plugin",
    "id": "The id of the Property sheet",
    "name": "The name of the plugin",
    "properties": [{
      "allowedValuePropSheetDef": "The id of the
PropSheetDef containing the allowed values for the
property",
      "allowedValues": [{
        "id": "The id of the allowed value",
        "label": "The label of the allowed value",
        "value": "The value of the allowed value"
      }],
      "default-value": "The default value of the
property",
      "description": "The description of the property",
      "hidden": "Whether or not the property is hidden",
      "id": "The id of the property",
      "label": "A label for this property shown when
prompting users for a value",
      "name": "The name of the property",
      "required": "Whether or not the value of the
property is required",
      "secure": "Whether or not the value of the
property is secure",
      "type": "The type of the property",
      "value": "The value of the property. This is only
used for properties on non-template objects (project, but
not project template, etc)"
    }]
  }],
  "type": "The type of the command",
  "validationScript": "The validation script of the
command",
  "validationScriptLanguage": "The language of the
validation script"
}],
"description": "The description of the plugin",
"hash": "The hash value of the plugin",
"id": "The id of the plugin",
"location": "The URL that can be used to get the JSON
representation of the plugin",
"name": "The name of the plugin",
"pluginId": "The Plug-in ID of the plugin",
"pluginVersion": "The version of the plugin",
"propSheetDefs": [{
  "description": "The description of the plugin",
  "id": "The id of the Property sheet",
  "name": "The name of the plugin",
  "properties": [{
    "allowedValuePropSheetDef": "The id of the
PropSheetDef containing the allowed values for the
property",
    "allowedValues": [{
      "id": "The id of the allowed value",
      "label": "The label of the allowed value",
      "value": "The value of the allowed value"
```

```
      }],
      "default-value": "The default value of the property",
      "description": "The description of the property",
      "hidden": "Whether or not the property is hidden",
      "id": "The id of the property",
      "label": "A label for this property shown when
  prompting users for a value",
      "name": "The name of the property",
      "required": "Whether or not the value of the
  property is required",
      "secure": "Whether or not the value of the property
  is secure",
      "type": "The type of the property",
      "value": "The value of the property. This is only
  used for properties on non-template objects (project, but
  not project template, etc)"
    }]
  }],
  "tag": "The tag of the plugin",
  "type": "The type of the plugin"
}
```

Update the integration

Request

```
PUT https://{hostname}:{port}
  /build/rest2/plugins{id}/integrations{id}
```

Table 233. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up plugin. This is the ID or name of the plugin
id	string	true	The identifier to use when looking up an integration. This is ID of the integration

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "propName": "propValue, if having multi properties,
  repeat this. If the propValue is an array, please use
  JSONArray format.",
  "propSheetName": "The name of the plugin"
}
```

Upload a plugin

Request

```
PUT https://{hostname}:{port}
  /build/rest2/plugins
```

Upload a plugin and allow downgrading if necessary

Request

```
PUT https://{hostname}:{port}
  /build/rest2/plugins/allowdowngrade
```

preferences resource

preferencesavailableLocales GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/preferencesavailableLocales
```

preflight resource

/preflight/lookupArtifactSets POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/preflight/lookupArtifactSets
```

/preflight/request POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/preflight/request
```

/preflight/request/{requestId} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/preflight/request/{requestId}
```

Table 234. URL parameters

Parameter	Type	Required	Description
requestId	long	true	

/preflight/build/{buildLifeId} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/preflight/build/{buildLifeId}
```

Table 235. URL parameters

Parameter	Type	Required	Description
buildLifeId	long	true	

processRequests resource

/processRequests POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processRequests
```

/processRequests{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/processRequests{id}
```

Table 236. URL parameters

Parameter	Type	Required	Description
id	long	true	

processTemplates resource

Resources for working with process templates.

/processTemplates/{id}/workflow GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow
```

Table 237. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/processTemplates/{id}/workflow/graph GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph
```

Table 238. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/processTemplates/{id}/workflow/graph POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph
```

Table 239. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/processTemplates/{id}/workflow/graph PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph
```

Table 240. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/processTemplates/{id}/workflow/graph/insert POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/insert
```

Table 241. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/processTemplates/{id}/workflow/graph/{id} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}
```

Table 242. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/processTemplates/{id}/workflow/graph/{id}/iterate DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}/iterate
```

Table 243. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/processTemplates/{id}/workflow/graph/{id}/iterate POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}/iterate
```

Table 244. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/processTemplates/{id}/workflow/graph/{id}/iterate/property POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 245. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/processTemplates/{id}/workflow/graph/{id}/iterate/property GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 246. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/processTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName}
```

Table 247. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.
propertyName	string	true	The identifier of a job config.

Create a new property by copy on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/copyproperty
```

Table 248. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Create a process template

Request

```
POST http://{hostname}:{port}
  /rest2/processTemplates
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }]
}
```

Delete a job in the process definition.

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/workflow/graph/{id}
```

Table 249. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

Delete a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}
```

Table 250. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Delete a specified name property on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/property/{name}
```

Table 251. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
name	string	true	The name of the property

Delete the artifact on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/artifacts/{id}
```

Table 252. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	long	true	The id of the artifact

Delete the resource on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/resources/{seq}
```

Table 253. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
seq	int	true	The sequence of the resource

Get a list of processes that use the specified process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/processes?{parameters}
```

Table 254. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process templates to return per page.
pageNumber	string	false	The page of process templates that should be returned.

Table 254. Request parameters (continued)

Parameter	Type	Required	Description
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'asc' or 'desc'.

Table 255. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process using this process
  template is active",
  "description": "The description of the process using
  this process template",
  "id": "The ID of the process using this process template",
  "name": "The name of the process using this process
  template",
  "project": "The name of the project to which the process
  using this process template belongs"
}]
```

Get all artifacts on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/artifacts?{parameters}
```

Table 256. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template artifacts to return per page.
pageNumber	string	false	The page of process template artifacts that should be returned.

Table 257. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all properties on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/properties?{parameters}
```

Table 258. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template properties to return per page.
pageNumber	string	false	The page of process template properties that should be returned.

Table 259. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/resources
```

Table 260. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get avaiable resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/resourceOptions/{id}
```

Table 261. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	int	true	The id of the selected resource

Get information about a process template

Return a JSON representation of a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates/{id}
```

Table 262. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
```

```
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Get information about all process templates

Return a JSON representation of all process templates.

Request

```
GET https://{hostname}:{port}
  /build/rest2/processTemplates?{parameters}
```

Table 263. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of process templates to return per page.
pageNumber	int	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'ASC' or 'DESC'.

Table 263. Request parameters (continued)

Parameter	Type	Required	Description
filterValue_name	string	false	The name pattern of the process template to search for.
filterValue_type	string	false	The type of the process template to search for. Possible values are Any/Build/Secondary

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
```

```
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}]
```

Update a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}
```

Table 264. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
```

```
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Update artifacts on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/artifacts
```

Table 265. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update properties on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/properties
```

Table 266. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update resources on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/processTemplates/{id}/resources
```

Table 267. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

projects resource

Get project details

Returns the details of the project

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}
```

Table 268. URL parameters

Parameter	Type	Required	Description
id	string	true	Project id or project name

/projects/{id}/buildProcesses/{id}/buildLives GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives?{parameters}
```

Table 269. Request parameters

Parameter	Type	Required	Description
includeInactive	string	false	

Table 270. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 270. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
preflightType	string	false	You can filter the build lives based on prefight type using the following values: • NOT_PREFLIGHT • ONLY_PREFLIGHT • ALL

/projects/{id}/buildProcesses/{id}/buildLives/latest GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest
```

Table 271. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies
```

Table 272. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies
```

Table 273. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifact-configs GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifact-configs
```

Table 274. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies-xml GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies-xml
```

Table 275. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 275. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets
```

Table 276. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}
```

Table 277. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
set	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts?{parameters}
```

Table 278. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 279. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
set	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts/{path}
```

Table 280. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
set	string	true	

Table 280. URL parameters (continued)

Parameter	Type	Required	Description
path	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories
```

Table 281. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}
```

Table 282. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts GET

Request

```
GET https://{hostname}:{port}

 /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts?{parameters}
```

Table 283. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 284. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts/{path}
```

Table 285. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

Table 285. URL parameters (continued)

Parameter	Type	Required	Description
path	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}
```

Table 286. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies
```

Table 287. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies
```

Table 288. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifact-configs GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifact-configs
```

Table 289. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies-xml GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies-xml
```

Table 290. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets
```

Table 291. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}
```

Table 292. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 292. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
set	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts?{parameters}
```

Table 293. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 294. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
set	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts/{path}
```

Table 295. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
set	string	true	
path	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories
```

Table 296. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}
```

Table 297. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts GET

Request

```
GET https://{hostname}:{port}

 /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts?{parameters}
```

Table 298. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of artifacts to return per page.
pageNumber	string	false	The page of artifacts that should be returned.

Table 299. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
id	string	true	

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts/{path} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts/{path}
```

Table 300. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
id	string	true	
path	string	true	

/projects/{id}/buildProcesses/{id}/dependencies GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies
```

Table 301. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/dependencies PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies
```

Table 302. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/dependencies POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies
```

Table 303. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/dependencies/complex GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies/complex
```

Table 304. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/dependencies/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies/{id}
```

Table 305. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/dependencies/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies/{id}
```

Table 306. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/dependencies/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencies/{id}
```

Table 307. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/triggers GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/triggers
```

Table 308. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/triggers POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/triggers
```

Table 309. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/triggers/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/triggers/{id}
```

Table 310. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/triggers/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/triggers/{id}
```

Table 311. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/triggers/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/triggers/{id}
```

Table 312. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

/projects/{id}/buildProcesses/{id}/successRateReport GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/successRateReport?{parameters}
```

Table 313. Request parameters

Parameter	Type	Required	Description
time	string	false	
type	string	false	

Table 314. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/buildProcesses/{id}/successRateReport PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/successRateReport
```

Table 315. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

/projects/{id}/secondaryProcesses/{id}/dashboardActivity GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/dashboardActivity
```

Table 316. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

/projects/{id}/secondaryProcesses/{id}/triggers GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/triggers
```

Table 317. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

/projects/{id}/secondaryProcesses/{id}/triggers POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/triggers
```

Table 318. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

/projects/{id}/secondaryProcesses/{id}/triggers/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/triggers/{id}
```

Table 319. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process
id	string	true	

/projects/{id}/secondaryProcesses/{id}/triggers/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/triggers/{id}
```

Table 320. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 320. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process
id	string	true	

/projects/{id}/secondaryProcesses/{id}/triggers/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/triggers/{id}
```

Table 321. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process
id	string	true	

/projects/{id}/notificationSubscriptions GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/notificationSubscriptions
```

Table 322. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

/projects/{id}/recentProcessActivity GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/recentProcessActivity
```

Table 323. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

/projects/{id}/latestStatus GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/latestStatus
```

Table 324. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

/projects/{id}/latestBuilds GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/latestBuilds
```

Table 325. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

/projects/{id}/sourceChanges GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/sourceChanges?{parameters}
```

Table 326. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of source changes to return per page.
pageNumber	string	false	The page of source changes that should be returned. The first page is 1
from	string	false	The field to use when getting the source changes from this date.
to	string	false	The field to use when getting the source chagnes to this date.

Table 327. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

/projects/{id}/sourceChangesByUser GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/sourceChangesByUser?{parameters}
```

Table 328. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of source changes to return per page.
pageNumber	string	false	The page of source changes that should be returned. The first page is 1
from	string	false	The field to use when getting the source changes from this date.
to	string	false	The field to use when getting the source chagnes to this date.

Table 329. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Create a build process

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses
```

Table 330. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The build process description",
  "name": "The name of the build process",
  "notificationScheme": "The notification scheme of the
  build process",
  "priority": "The priority of the build process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "teams": "The array of team names that have access to
  the build process",
  "template": "The process template for this build process",
```

```
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Create a label.

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/labels
```

Table 331. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "label": "The name of the label.",
  "labelDependencies": "Determines which dependencies to
  which the label should also be applied. This is not
  required. The value should be 'NONE', 'DIRECT' or 'ALL'."
}
```

Create a label.

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/labels
```

Table 332. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "label": "The name of the label.",
  "labelDependencies": "Determines which dependencies to
  which the label should also be applied. This is not
  required. The value should be 'NONE', 'DIRECT' or 'ALL'."
}
```

Create a project

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The project description",
  "name": "The name of the project",
  "tags": "The array of tag names of the project",
  "teams": "The array of team names of the project",
  "template": "The project template for this project",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Create a secondary process

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses
```

Table 333. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The secondary process description",
  "name": "The name of the secondary process",
  "notificationScheme": "The notification scheme of the
  secondary process",
  "priority": "The priority of the secondary process",
  "properties": [{
    "allowed-values": "The values users are allowed to
```

```
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "teams": "The array of team names that have access to
  the secondary process",
  "template": "The process template for this secondary
  process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Create a source config

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/sourceConfigs
```

Table 334. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "ignoreRepositoryTrigger": "Whether or not the source
  config should ignore repository triggers",
  "name": "The name of the source config",
  "respository": "The repository that the source config
  will use",
  "templateId": "The source config template ID for this
  source config",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Create a status.

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/statuses
```

Table 335. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "externalMessage": "The message of the status if the
  status is created by an external source.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "status": "The name of the status."
}
```

Create a status.

Request

```
POST https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/statuses
```

Table 336. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "externalMessage": "The message of the status if the
  status is created by an external source.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "status": "The name of the status."
}
```

Delete a specified name property in process propertes

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/processes/{id}/properties/{name}
```

Table 337. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a process. This can be the name or ID of the process
name	string	true	The name of the property

Delete a specified name property in process propertes

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/properties/{name}
```

Table 338. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
name	string	true	The name of the property

Delete a specified name property in process propertes

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/properties/{name}
```

Table 339. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process
name	string	true	The name of the property

Delete the build process

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}
```

Table 340. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 340. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Delete the project

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}
```

Table 341. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Delete the secondary process

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}
```

Table 342. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

Delete the source config

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/sourceConfigs/{id}
```

Table 343. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config

Get a summary of all the findings in the source analytic report

Return a JSON representation of the summary of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries/{name}/summaries
```

Table 344. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get a summary of all the findings in the source analytic report

Return a JSON representation of the summary of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries/{name}/summaries
```

Table 345. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get activities shown on the process dashboard

Return a JSON representation of the activities on the process dashboard

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dashboardActivity
```

Table 346. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get all the source analytic reports

Return a JSON representation of all source analytic reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics
```

Table 347. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic report.",
    "count": "The count of findings in the source analytic
  report.",
    "id": "The id of the source analytic report.",
    "location": "The REST location to get the source
  analytic report.",
    "name": "The name of the source analytic report.",
    "type": "The type of the source analytic report.",
    "url": "The URL of the source analytic report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic reports

Return a JSON representation of all source analytic reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics
```

Table 348. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 348. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic report.",
    "count": "The count of findings in the source analytic
  report.",
    "id": "The id of the source analytic report.",
    "location": "The REST location to get the source
  analytic report.",
    "name": "The name of the source analytic report.",
    "type": "The type of the source analytic report.",
    "url": "The URL of the source analytic report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic summary reports

Return a JSON representation of all source analytic summary reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries
```

Table 349. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
```

```
    "count": "The count of findings in the source analytic
  summary report.",
    "id": "The id of the source analytic summary report.",
    "location": "The REST location to get the source
  analytic summary report.",
    "name": "The name of the source analytic summary
  report.",
    "type": "The type of the source analytic summary
  report.",
    "url": "The URL of the source analytic summary report."
  }],
  "totalRecords": "The number of reports."
}
```

Get all the source analytic summary reports

Return a JSON representation of all source analytic summary reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries
```

Table 350. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
    "count": "The count of findings in the source analytic
  summary report.",
    "id": "The id of the source analytic summary report.",
    "location": "The REST location to get the source
  analytic summary report.",
    "name": "The name of the source analytic summary
  report.",
    "type": "The type of the source analytic summary
  report.",
    "url": "The URL of the source analytic summary report."
  }],
  "totalRecords": "The number of reports."
}
```

Get dependencies of the build process

Return a XML representation of the dependencies of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependenciesXml
```

Table 351. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get dependency summaries of the build process

Return a JSON representation of the dependency summaries of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencySummaries
```

Table 352. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get general dependency configuration of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencyConfig
```

Table 353. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get information about all code coverage reports

Return a JSON representation of all code coverage reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/codeCoverage
```

Table 354. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "branchPercentage": "The branch percentage of the code
  coverage report.",
    "buildLifeId": "The id of the build life that contains
  the code coverage report.",
    "groups": [{
      "branchPercentage": "The branch percentage of the
  group in the code coverage report.",
      "complexity": "The complexity of the group in the
  code coverage report.",
      "linePercentage": "The line percentage of the group
  in the code coverage report.",
      "methodPercentage": "The method percentage of the
  group in the code coverage report.",
      "name": "The name of the group in the code coverage
  report."
    }],
    "linePercentage": "The line percentage of the code
  coverage report.",
    "methodPercentage": "The method percentage of the code
  coverage report.",
```

```
    "name": "The name of the code coverage report.",
    "type": "The type of the code coverage report."
  }],
  "totalReports": "The number of the code coverage reports."
}
```

Get information about all code coverage reports

Return a JSON representation of all code coverage reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/codeCoverage
```

Table 355. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "branchPercentage": "The branch percentage of the code
  coverage report.",
    "buildLifeId": "The id of the build life that contains
  the code coverage report.",
    "groups": [{
      "branchPercentage": "The branch percentage of the
  group in the code coverage report.",
      "complexity": "The complexity of the group in the
  code coverage report.",
      "linePercentage": "The line percentage of the group
  in the code coverage report.",
      "methodPercentage": "The method percentage of the
  group in the code coverage report.",
      "name": "The name of the group in the code coverage
  report."
    }],
    "linePercentage": "The line percentage of the code
  coverage report.",
    "methodPercentage": "The method percentage of the code
  coverage report.",
    "name": "The name of the code coverage report.",
    "type": "The type of the code coverage report."
  }],
```

```
  "totalReports": "The number of the code coverage reports."
}
```

Get information about all issues.

Return a JSON representation of all issues.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/issues
```

Table 356. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "count": "The number of the issues.",
  "issues": [{
    "id": "The id of the issue.",
    "name": "The name of the issue."
  }]
}
```

Get information about all issues.

Return a JSON representation of all issues.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/issues
```

Table 357. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Table 357. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "count": "The number of the issues.",
  "issues": [{
    "id": "The id of the issue.",
    "name": "The name of the issue."
  }]
}
```

Get information about all labels

Return a JSON Array representation of all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/labels
```

Table 358. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "assignedDate": "The assigned date of the label.",
  "assignedDateFormat": "The format of the assigned date.",
  "buildLifeId": "The id of the build life.",
  "label": "The name of the label.",
  "labelId": "The id of the label.",
  "user": "The user that applied the label."
}]
```

Get information about all labels

Return a JSON Array representation of all labels.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/labels
```

Table 359. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "assignedDate": "The assigned date of the label.",
  "assignedDateFormat": "The format of the assigned date.",
  "buildLifeId": "The id of the build life.",
  "label": "The name of the label.",
  "labelId": "The id of the label.",
  "user": "The user that applied the label."
}]
```

Get information about all of the findings in the source analytic report

Return a JSON representation of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics/{id}/findings
```

Table 360. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get information about all of the findings in the source analytic report

Return a JSON representation of all the findings in the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics/{id}/findings
```

Table 361. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "records": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "totalRecords": "The number of findings."
}
```

Get information about all source change sets

Return a JSON Array representation of all source change sets.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/sourceChange
```

Table 362. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "comment": "The comment of the source change set.",
  "commitDate": "The commit date of the source change set.",
  "id": "The id of the source change set.",
  "paths": [{
    "path": "The path of the change.",
    "type": "The type of the change."
  }],
  "user": "The user name who submitted the source change
  set.",
  "version": "The version of the source change set."
}]
```

Get information about all source change sets

Return a JSON Array representation of all source change sets.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/sourceChange
```

Table 363. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Table 363. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "comment": "The comment of the source change set.",
  "commitDate": "The commit date of the source change set.",
  "id": "The id of the source change set.",
  "paths": [{
    "path": "The path of the change.",
    "type": "The type of the change."
  }],
  "user": "The user name who submitted the source change
  set.",
  "version": "The version of the source change set."
}]
```

Get information about all source configs

Return a JSON representation of all source configs.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/sourceConfigs
```

Table 364. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "id": "The ID of the source config",
  "ignoreRepositoryTrigger": "Whether or not the source
  config should ignore repository triggers",
  "inTemplateCompliance": "Whether or not the source
  config is in compliance with its template",
  "location": "The URL that can be used to get the JSON
  representation of the source config",
  "name": "The name of the source config",
  "respository": "The repository that the source config
```

```
  will use",
  "templateId": "The source config template ID for this
  source config",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}]
```

Get information about all statuses

Return a JSON Array representation of all statuses.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/statuses
```

Table 365. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "buildLifeId": "The id of the build life.",
  "dateAssigned": "The assigned date of the status.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "message": "The message of the status if the status is
  created by a user or external source.",
  "origin": "The origin of the status.",
  "status": "The status.",
  "statusColor": "The color of the status.",
  "statusName": "The name of the status.",
  "user": "The name of the user if the status is created
  by a user."
}]
```

Get information about all statuses

Return a JSON Array representation of all statuses.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/statuses
```

Table 366. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "buildLifeId": "The id of the build life.",
  "dateAssigned": "The assigned date of the status.",
  "jobTraceId": "The id of the job trace if the status is
  created by a job.",
  "message": "The message of the status if the status is
  created by a user or external source.",
  "origin": "The origin of the status.",
  "status": "The status.",
  "statusColor": "The color of the status.",
  "statusName": "The name of the status.",
  "user": "The name of the user if the status is created
  by a user."
}]
```

Get information about all test reports

Return a JSON representation of all test reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/tests
```

Table 367. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
```

```
    "failure": "The number of failures in the test report.",
    "name": "The name of the test report.",
    "success": "The number of successes in the test
  report.",
    "successPercent": "The success percent of the test
  report.",
    "suites": [{
      "failure": "The number of failures in the test
  suite.",
      "name": "The name of the test suite.",
      "success": "The number of successes in the test
  suite.",
      "successPercent": "The success percent of the test
  suite."
    }]
  }],
  "successPercent": "The success percent of all the test
  reports."
}
```

Get information about all test reports

Return a JSON representation of all test reports.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/tests
```

Table 368. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "reports": [{
    "failure": "The number of failures in the test report.",
    "name": "The name of the test report.",
    "success": "The number of successes in the test
  report.",
    "successPercent": "The success percent of the test
  report.",
    "suites": [{
      "failure": "The number of failures in the test
  suite.",
```

```
      "name": "The name of the test suite.",
      "success": "The number of successes in the test
  suite.",
      "successPercent": "The success percent of the test
  suite."
    }]
  }],
  "successPercent": "The success percent of all the test
  reports."
}
```

Get information about build activity for a specific build life

Return a JSON representation of the build activity

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/activity/{id}
```

Table 369. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	int	true	The ID of the build life

Get information about changeset for a specific userName

Return a JSON representation of the changeset

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/sourceChanges/changeset/{userName}?{parameters}
```

Table 370. Request parameters

Parameter	Type	Required	Description
from	string	false	The field to use when getting the source changes from this date.
to	string	false	The field to use when getting the source chagnes to this date.

Table 371. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
userName	string	true	The userName of the changeset

Get information about changeset for a specific userName

Return a JSON representation of the changeset

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/sourceChangesByUser/changeset/{userName}?{parameters}
```

Table 372. Request parameters

Parameter	Type	Required	Description
from	string	false	The field to use when getting the source changes from this date.
to	string	false	The field to use when getting the source chagnes to this date.

Table 373. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
userName	string	true	The userName of the changeset

Get information about queried build processes

Return a JSON representation of build processes.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses?{parameters}
```

Table 374. Request parameters

Parameter	Type	Required	Description
nameLike	string	false	The pattern to use when looking up the process by name
pageNumber	int	false	The page of build processes that should be returned
pageLineCount	int	false	The number of build processes to return per page

Table 375. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the build process is active",
  "description": "The build process description",
  "id": "The ID of the build process",
  "location": "The URL that can be used to get the JSON
  representation of the build process",
  "name": "The name of the build process",
  "notificationScheme": "The notification scheme of the
  build process",
  "priority": "The priority of the build process",
  "projectId": "The ID of the project that contains the
  build process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
```

```
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}]
```

Get information about queried processes

Return a JSON representation of processes.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/processes?{parameters}
```

Table 376. Request parameters

Parameter	Type	Required	Description
modifiedSince	int	false	The time in millisecond
nameLike	string	false	The pattern to use when looking up the process by name
pageNumber	int	false	The page of processes that should be returned
pageLineCount	int	false	The number of processes to return per page

Table 377. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the build process is active",
  "description": "The build process description",
  "id": "The ID of the build process",
  "location": "The URL that can be used to get the JSON
  representation of the build process",
  "name": "The name of the build process",
  "notificationScheme": "The notification scheme of the
  build process",
```

```
  "priority": "The priority of the build process",
  "projectId": "The ID of the project that contains the
  build process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}]
```

Get information about queried projects

Return a JSON representation of queried projects.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects?{parameters}
```

Table 378. Request parameters

Parameter	Type	Required	Description
nameLike	string	false	The pattern to use when looking up the project by name.
pageNumber	string	false	The page of projects that should be returned.

Table 378. Request parameters (continued)

Parameter	Type	Required	Description
pageLineCount	string	false	The number of projects to return per page.
activeFilter	string	false	The active filter of projects. Possible values are active/inactive/all.

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the project is active.",
  "description": "The project description",
  "id": "The ID of the project",
  "location": "The URL that can be used to get the JSON
  representation of the project",
  "name": "The name of the project",
  "tags": "The array of tag names of the project",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}]
```

Get information about queried secondary processes

Return a JSON representation of secondary processes.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses
```

Table 379. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the secondary process is
  active",
  "description": "The secondary process description",
  "id": "The ID of the secondary process",
  "location": "The URL that can be used to get the JSON
  representation of the secondary process",
  "name": "The name of the secondary process",
  "notificationScheme": "The notification scheme of the
  secondary process",
  "priority": "The priority of the secondary process",
```

```
  "projectId": "The ID of the project containing the
  secondary process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}]
```

Get information about the build process

Return a JSON representation of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}
```

Table 380. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 380. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the build process is active",
  "description": "The build process description",
  "id": "The ID of the build process",
  "location": "The URL that can be used to get the JSON
  representation of the build process",
  "name": "The name of the build process",
  "notificationScheme": "The notification scheme of the
  build process",
  "priority": "The priority of the build process",
  "projectId": "The ID of the project that contains the
  build process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Get information about the project

Return a JSON representation of the project

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}
```

Table 381. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the project is active.",
  "description": "The project description",
  "id": "The ID of the project",
  "location": "The URL that can be used to get the JSON
  representation of the project",
  "name": "The name of the project",
  "tags": "The array of tag names of the project",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Get information about the secondary process

Return a JSON representation of the secondary process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}
```

Table 382. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the secondary process is
  active",
  "description": "The secondary process description",
  "id": "The ID of the secondary process",
  "location": "The URL that can be used to get the JSON
  representation of the secondary process",
  "name": "The name of the secondary process",
  "notificationScheme": "The notification scheme of the
  secondary process",
  "priority": "The priority of the secondary process",
  "projectId": "The ID of the project containing the
  secondary process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Get information about the source analytic report

Return a JSON representation of the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics/{id}
```

Table 383. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic report.",
  "count": "The count of findings in the source analytic
  report.",
  "findings": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic report.",
  "location": "The REST location to get the source
  analytic report.",
  "name": "The name of the source analytic report.",
  "type": "The type of the source analytic report.",
  "url": "The URL of the source analytic report."
}
```

Get information about the source analytic report

Return a JSON representation of the source analytic report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics/{id}
```

Table 384. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 384. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	
id	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic report.",
  "count": "The count of findings in the source analytic
  report.",
  "findings": [{
    "description": "The description of the finding.",
    "file": "The file containing the finding.",
    "id": "The id of the finding.",
    "line": "The line number of the finding in the file.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "status": "The status of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic report.",
  "location": "The REST location to get the source
  analytic report.",
  "name": "The name of the source analytic report.",
  "type": "The type of the source analytic report.",
  "url": "The URL of the source analytic report."
}
```

Get information about the source analytic summary report

Return a JSON representation of the source analytic summary report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries/{name}
```

Table 385. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

Table 385. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
  "count": "The count of findings in the source analytic
  summary report.",
  "findings": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic summary report.",
  "location": "The REST location to get the source
  analytic summary report.",
  "name": "The name of the source analytic summary report.",
  "type": "The type of the source analytic summary report.",
  "url": "The URL of the source analytic summary report."
}
```

Get information about the source analytic summary report

Return a JSON representation of the source analytic summary report.

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries/{name}
```

Table 386. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Table 386. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	
name	string	true	The identifier to use when looking up a source analytic report.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "buildLifeId": "The id of the build life that contains
  the source analytic summary report.",
  "count": "The count of findings in the source analytic
  summary report.",
  "findings": [{
    "count": "The count of the finding.",
    "description": "The description of the finding.",
    "name": "The name of the finding.",
    "severity": "The severity of the finding.",
    "url": "The url of the finding."
  }],
  "id": "The id of the source analytic summary report.",
  "location": "The REST location to get the source
  analytic summary report.",
  "name": "The name of the source analytic summary report.",
  "type": "The type of the source analytic summary report.",
  "url": "The URL of the source analytic summary report."
}
```

Get information about the source config

Return a JSON representation of the source config

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/sourceConfigs/{id}
```

Table 387. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Table 387. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "id": "The ID of the source config",
  "ignoreRepositoryTrigger": "Whether or not the source
  config should ignore repository triggers",
  "inTemplateCompliance": "Whether or not the source
  config is in compliance with its template",
  "location": "The URL that can be used to get the JSON
  representation of the source config",
  "name": "The name of the source config",
  "respository": "The repository that the source config
  will use",
  "templateId": "The source config template ID for this
  source config",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Get paginated build activity information

Return a JSON representation of the activities in pages

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/activity/{pageSize}/{page}
```

Table 388. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
pageSize	int	true	The number of activities to return per page
page	int	true	The page of activities that should be returned

Get paginated build activity information

Return a JSON representation of the activities in pages and in a specified order

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/activity?{parameters}
```

Table 389. Request parameters

Parameter	Type	Required	Description
numPerPage	int	false	The number of activities to return per page
pageNumber	int	false	The page of activities that should be returned
sortType	string	false	The order to use when sorting the activities. This would either be 'asc' or 'desc'.

Table 390. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get properties of the build process

Return a JSON representation of the properties of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/runtimeProperties
```

Table 391. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get properties on a process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/processes/{id}/properties?{parameters}
```

Table 392. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process properties to return per page.
pageNumber	string	false	The page of process properties that should be returned.

Table 393. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a process. This can be the name or ID of the process

Get properties on a process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/properties?{parameters}
```

Table 394. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process properties to return per page.
pageNumber	string	false	The page of process properties that should be returned.

Table 395. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Get properties on a process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/properties?{parameters}
```

Table 396. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process properties to return per page.
pageNumber	string	false	The page of process properties that should be returned.

Table 397. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

Get status summaries of the build process

Return a JSON representation of the status summaries of the build process

Request

```
GET https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/statusSummaries
```

Table 398. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Update general dependency configuration of the build process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/dependencyConfig?{parameters}
```

Table 399. Request parameters

Parameter	Type	Required	Description
conflictStrategy	string	false	Dependency conflict strategy of the build process. Options are: FAIL, FAVOR_NEW, FAVOR_OLD
triggerOnly	boolean	false	Whether or not the process dependency settings should only be used for triggering builds

Table 400. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Update or create property on a process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/processes/{id}/properties
```

Table 401. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a process. This can be the name or ID of the process

Update or create property on a process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/properties
```

Table 402. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

Update or create property on a process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}/properties
```

Table 403. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

Update the build process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}
```

Table 404. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the build process is active",
  "description": "The build process description",
  "name": "The name of the build process",
  "notificationScheme": "The notification scheme of the
  build process",
  "priority": "The priority of the build process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
```

```
    "value": "The value of the property"
  }]
}
```

Update the project

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}
```

Table 405. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the project is active.",
  "description": "The project description",
  "name": "The name of the project",
  "tags": "The array of tag names of the project",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Update the secondary process

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/secondaryProcesses/{id}
```

Table 406. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a secondary process. This can be the name or ID of the process

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the secondary process is
  active",
  "description": "The secondary process description",
  "name": "The name of the secondary process",
  "notificationScheme": "The notification scheme of the
  secondary process",
  "priority": "The priority of the secondary process",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "skipPreProcessing": "Whether or not to skip pre-
  processing when running an unforced build for the process",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

Update the source config

Request

```
PUT https://{hostname}:{port}
  /build/rest2/projects/{id}/buildProcesses/{id}/sourceConfigs/{id}
```

Table 407. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project. This can be the name or ID of the project
id	string	true	The identifier to use when looking up a build process. This can be the name or ID of the process
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "ignoreRepositoryTrigger": "Whether or not the source
  config should ignore repository triggers",
  "name": "The name of the source config",
  "respository": "The repository that the source config
  will use",
  "templatedProperties": [{
    "name": "The name of the property",
    "value": "The value of the property"
  }]
}
```

radiator resource

/radiator GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/radiator
```

/radiator/{id} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/radiator/{id}
```

Table 408. URL parameters

Parameter	Type	Required	Description
id	long	true	

/radiator/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/radiator/{id}
```

Table 409. URL parameters

Parameter	Type	Required	Description
id	long	true	

repo resource

/repo POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repo
```

/repo/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repo/{id}
```

Table 410. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repo/{id}/{hash} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repo/{id}/{hash}
```

Table 411. URL parameters

Parameter	Type	Required	Description
id	string	true	

Table 411. URL parameters (continued)

Parameter	Type	Required	Description
hash	string	true	

/repo/{id}/byLabel/{label} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repo/{id}/byLabel/{label}
```

Table 412. URL parameters

Parameter	Type	Required	Description
id	string	true	
label	string	true	

/repo/{id}/{hash}/label/{label} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repo/{id}/{hash}/label/{label}
```

Table 413. URL parameters

Parameter	Type	Required	Description
id	string	true	
label	string	true	
hash	string	true	

/repo/{id}/{hash}/copyToServer POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repo/{id}/{hash}/copyToServer
```

Table 414. URL parameters

Parameter	Type	Required	Description
id	string	true	
hash	string	true	

/repo/{id}/{hash}/copyToUrbanCodeDeploy POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repo/{id}/{hash}/copyToUrbanCodeDeploy
```

Table 415. URL parameters

Parameter	Type	Required	Description
id	string	true	
hash	string	true	

reporting resource

/reporting/tables GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/tables
```

/reporting/tables POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/reporting/tables
```

/reporting/tables/{name} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/tables/{name}
```

Table 416. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/tables/{name} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/reporting/tables/{name}
```

Table 417. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/tables/{name}/expanded GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/tables/{name}/expanded
```

Table 418. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/tables/{name}/related GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/tables/{name}/related
```

Table 419. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/dimensions GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/dimensions
```

/reporting/dimensions POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/reporting/dimensions
```

/reporting/dimensions/{name} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/dimensions/{name}
```

Table 420. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/dimensions/{name}/expanded GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/dimensions/{name}/expanded
```

Table 421. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/tablesAndDimensions GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/tablesAndDimensions
```

/reporting/query POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/reporting/query
```

/reporting/query/download GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/query/download?{parameters}
```

Table 422. Request parameters

Parameter	Type	Required	Description
query	string	false	
filename	string	false	

/reporting/reports GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/reports
```

/reporting/reports POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/reporting/reports
```

/reporting/reports/{name} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/reporting/reports/{name}
```

Table 423. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/reports/{name} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/reporting/reports/{name}
```

Table 424. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/reports/{name} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reporting/reports/{name}?{parameters}
```

Table 425. Request parameters

Parameter	Type	Required	Description
download	boolean	false	

Table 426. URL parameters

Parameter	Type	Required	Description
name	string	true	

/reporting/reports/{name}/output POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/reporting/reports/{name}/output
```

Table 427. URL parameters

Parameter	Type	Required	Description
name	string	true	

reports resource

/reports/tests/projects GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/tests/projects
```

/reports/tests/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/tests/projects/{id}?{parameters}
```

Table 428. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 429. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/tests/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/tests/projects/{id}/build-processes/{id}?{parameters}
```

Table 430. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 431. URL parameters

Parameter	Type	Required	Description
id	string	true	

Table 431. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	

/reports/builds/projects GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/builds/projects
```

/reports/builds/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/builds/projects/{id}
```

Table 432. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/builds/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/builds/projects/{id}/build-processes/{id}
```

Table 433. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/reports/commits/projects GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/commits/projects
```

/reports/commits/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/commits/projects/{id}?{parameters}
```

Table 434. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 435. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/commits/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/commits/projects/{id}/build-processes/{id}?{parameters}
```

Table 436. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 437. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/reports/issues/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/issues/projects/{id}?{parameters}
```

Table 438. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 439. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/issues/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/issues/projects/{id}/build-processes/{id}?{parameters}
```

Table 440. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 441. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/reports/coverage/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/coverage/projects/{id}?{parameters}
```

Table 442. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 443. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/coverage/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/coverage/projects/{id}/build-processes/{id}?{parameters}
```

Table 444. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 445. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

/reports/analytics/projects/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/analytics/projects/{id}?{parameters}
```

Table 446. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 447. URL parameters

Parameter	Type	Required	Description
id	string	true	

/reports/analytics/projects/{id}/build-processes/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/reports/analytics/projects/{id}/build-processes/{id}?{parameters}
```

Table 448. Request parameters

Parameter	Type	Required	Description
startDate	string	false	
endDate	string	false	

Table 449. URL parameters

Parameter	Type	Required	Description
id	string	true	
id	string	true	

repositories resource

/repositories GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repositories
```

/repositories POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repositories
```

/repositories/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repositories/{id}
```

Table 450. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repositories/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/repositories/{id}
```

Table 451. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repositories/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/repositories/{id}
```

Table 452. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repositories/{id}/trigger GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/repositories/{id}/trigger
```

Table 453. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repositories/{id}/trigger POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/repositories/{id}/trigger
```

Table 454. URL parameters

Parameter	Type	Required	Description
id	string	true	

/repositories/{id}/trigger DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/repositories/{id}/trigger
```

Table 455. URL parameters

Parameter	Type	Required	Description
id	string	true	

requestStatuses resource

/requestStatuses GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/requestStatuses
```

resource resource

resource/{resource}/teamMappings GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resource/{resource}/teamMappings
```

Table 456. URL parameters

Parameter	Type	Required	Description
resource	string	true	

resourceRole resource

resourceRole GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceRole
```

resourceRole POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/resourceRole
```

resourceRole/{resourceRole} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceRole/{resourceRole}
```

Table 457. URL parameters

Parameter	Type	Required	Description
resourceRole	string	true	

resourceRole/{resourceRole} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/resourceRole/{resourceRole}
```

Table 458. URL parameters

Parameter	Type	Required	Description
resourceRole	string	true	

resourceRole/{resourceRole} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/resourceRole/{resourceRole}
```

Table 459. URL parameters

Parameter	Type	Required	Description
resourceRole	string	true	

resourceType resource

resourceType GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceType
```

resourceType/{resourceType} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceType/{resourceType}
```

Table 460. URL parameters

Parameter	Type	Required	Description
resourceType	string	true	

resourceType/{resourceType}/actions GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceType/{resourceType}/actions
```

Table 461. URL parameters

Parameter	Type	Required	Description
resourceType	string	true	

resourceType/{resourceType}/resourceRoles GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/resourceType/{resourceType}/resourceRoles
```

Table 462. URL parameters

Parameter	Type	Required	Description
resourceType	string	true	

role resource

role GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/role
```

role POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/role
```

role/{role} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/role/{role}
```

Table 463. URL parameters

Parameter	Type	Required	Description
role	string	true	

role/{role} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/role/{role}
```

Table 464. URL parameters

Parameter	Type	Required	Description
role	string	true	

role/{role} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/role/{role}
```

Table 465. URL parameters

Parameter	Type	Required	Description
role	string	true	

role/{role}/actionMappings GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/role/{role}/actionMappings
```

Table 466. URL parameters

Parameter	Type	Required	Description
role	string	true	

role/{role}/actionMappings POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/role/{role}/actionMappings
```

Table 467. URL parameters

Parameter	Type	Required	Description
role	string	true	

role/{role}/actionMappings DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/role/{role}/actionMappings
```

Table 468. URL parameters

Parameter	Type	Required	Description
role	string	true	

search resource

/search/requests GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/requests?{parameters}
```

Table 469. Request parameters

Parameter	Type	Required	Description
pageNumber	int	false	The page of BuildLifeRequests that should be returned.
rowsPerPage	int	false	The number of BuildLifeRequests to return per page.
filterValue_status	string	false	The status name to search for.
filterValue_name	string	false	The project name pattern to search for.

/search/statuses GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/statuses?{parameters}
```

Table 470. Request parameters

Parameter	Type	Required	Description
pageNumber	int	false	The page of statusSummary that should be returned.
rowsPerPage	int	false	The number of statusSummary to return per page.
filterValue_status	string	false	The status name to search for.
filterValue_name	string	false	The project name pattern to search for.

/search/workflows GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/workflows?{parameters}
```

Table 471. Request parameters

Parameter	Type	Required	Description
pageNumber	int	false	The page of build life that should be returned.
rowsPerPage	int	false	The number of build life to return per page.
filterValue_processStatusName	string	false	The workflowStatus name to search for.
filterValue_name	string	false	The project name pattern to search for.
filterValue_workflowName	string	false	The workflow name pattern to search for.

/search/workflows/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/workflows/{id}?{parameters}
```

Table 472. Request parameters

Parameter	Type	Required	Description
pageNumber	int	false	The page of build life that should be returned.
rowsPerPage	int	false	The number of build life to return per page.
filterValue_processStatusName	string	false	The workflowStatus name to search for.
filterValue_name	string	false	The project name pattern to search for.

Table 473. URL parameters

Parameter	Type	Required	Description
id	string	true	

/search/projects GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/projects?{parameters}
```

Table 474. Request parameters

Parameter	Type	Required	Description
name	string	false	

Abort buildRequest

Request

```
POST https://{hostname}:{port}
  /build/rest2/search/requests?{parameters}
```

Table 475. Request parameters

Parameter	Type	Required	Description
requestId	string	false	The id of buildRequest to abort.

Deactivate a build life

Request

```
POST https://{hostname}:{port}
  /build/rest2/search/deactivateBuild
```

Delete a build life

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/search/deleteBuild
```

Delete buildLives

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/search/buildLives?{parameters}
```

Table 476. Request parameters

Parameter	Type	Required	Description
buildLifeIds	string	false	The buildLifeIds, which are separated by ','.

Delete buildRequests

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/search/requests?{parameters}
```

Table 477. Request parameters

Parameter	Type	Required	Description
requestIds	string	false	The buildRequestIds, which are separated by ','.

Get builds by label.

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/label?{parameters}
```

Table 478. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of build life to return per page.
pageNumber	int	false	The page of build life that should be returned.
orderField	string	false	The field to use when sorting the build life.
sortType	string	false	The order to use when sorting the build life. This would either be 'asc' or 'desc'.
filterValue_projectId	long	false	The filter value of project.
filterValue_labelId	long	false	The filter value of label.

Get the build by searching.

Request

```
GET https://{hostname}:{port}
  /build/rest2/search/build?{parameters}
```

Table 479. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of build life to return per page.
pageNumber	int	false	The page of build life that should be returned.
orderField	string	false	The field to use when sorting the build life.
sortType	string	false	The order to use when sorting the build life. This would either be 'asc' or 'desc'.
filterValue_buildLifeId	string	false	The filter value of build life.
filterValue_name	string	false	The filter value of project.
filterValue_statusId	string	false	The filter value of status.
filterValue_stamp	string	false	The filter value of stamp.

secondaryProcessTemplates resource

Resources for working with secondary process templates.

/secondaryProcessTemplates/{id}/workflow GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow
```

Table 480. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/secondaryProcessTemplates/{id}/workflow/graph GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph
```

Table 481. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/secondaryProcessTemplates/{id}/workflow/graph POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph
```

Table 482. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/secondaryProcessTemplates/{id}/workflow/graph PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph
```

Table 483. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/secondaryProcessTemplates/{id}/workflow/graph/insert POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/insert
```

Table 484. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

/secondaryProcessTemplates/{id}/workflow/graph/{id} POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}
```

Table 485. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate
```

Table 486. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate
```

Table 487. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 488. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property
```

Table 489. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.

/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName}
```

Table 490. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	string	true	The identifier of a job config.
propertyName	string	true	The identifier of a job config.

Create a new property by copy on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/copyproperty
```

Table 491. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Create a process template

Request

```
POST https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
```

```
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }]
}
```

Delete a job in the process definition.

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/workflow/graph/{id}
```

Table 492. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Table 492. URL parameters (continued)

Parameter	Type	Required	Description
id	string	true	The identifier of a job config.

Delete a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}
```

Table 493. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Delete a specified name property on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/property/{name}
```

Table 494. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
name	string	true	The name of the property

Delete the artifact on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/artifacts/{id}
```

Table 495. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	long	true	The id of the artifact

Delete the resource on a process template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/resources/{seq}
```

Table 496. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
seq	int	true	The sequence of the resource

Get a list of processes that use the specified process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/processes?{parameters}
```

Table 497. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process templates to return per page.
pageNumber	string	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'asc' or 'desc'.

Table 498. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process using this process
  template is active",
  "description": "The description of the process using
  this process template",
  "id": "The ID of the process using this process template",
  "name": "The name of the process using this process
  template",
  "project": "The name of the project to which the process
  using this process template belongs"
}]
```

Get all artifacts on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/artifacts?{parameters}
```

Table 499. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template artifacts to return per page.
pageNumber	string	false	The page of process template artifacts that should be returned.

Table 500. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all properties on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/properties?{parameters}
```

Table 501. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of process template properties to return per page.
pageNumber	string	false	The page of process template properties that should be returned.

Table 502. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get all resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/resources
```

Table 503. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Get avaiable resources on a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/resourceOptions/{id}
```

Table 504. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template
id	int	true	The id of the selected resource

Get information about a process template

Return a JSON representation of a process template

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}
```

Table 505. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
```

```
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Get information about all secondary process templates

Return a JSON representation of all secondary process templates.

Request

```
GET https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates?{parameters}
```

Table 506. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of process templates to return per page.
pageNumber	int	false	The page of process templates that should be returned.
orderField	string	false	The field to use when sorting the process templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the process templates. This would either be 'ASC' or 'DESC'.
filterValue_name	string	false	The name pattern of the process template to search for.
filterValue_type	string	false	The type of the process template to search for. Possible values are Any/Build/Secondary

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "id": "The ID of the process template",
  "location": "The URL that can be used to get the JSON
  representation of the process template",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}]
```

Update a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}
```

Table 507. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the process template is
  active.",
  "artifact-set-configuration": [{
    "artifactSet": "The name of the Artifact Sets",
    "baseDir": "The directory containing the artifact
  files",
    "excludes": "The patterns used to determine which
  files should be excluded from the artifact set even if
  they match the include patterns",
    "includes": "The patterns used to determine which
  files should be included in the artifact set"
  }],
  "description": "The process template description",
  "name": "The name of the process template",
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "type": "The type of the process template (Build or
  Secondary)"
}
```

Update artifacts on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/artifacts
```

Table 508. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update properties on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/properties
```

Table 509. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

Update resources on a process template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/secondaryProcessTemplates/{id}/resources
```

Table 510. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a process template. This can be the name or ID of the process template

security resource

/security/user GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/security/user
```

sourceTemplates resource

Resources for working with source config templates.

Create a new property by copy on a source config template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}/copyproperty
```

Table 511. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

Create a source config template

Request

```
POST https://{hostname}:{port}
  /build/rest2/sourceTemplates
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The source config template description",
  "name": "The name of thesource config template",
  "plugin": "The Plug-in ID of Source Plug-in",
  "plugin-properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
```

```
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }],
  "work-dir-script": "The id of working directory script"
}
```

Delete a source config template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}
```

Table 512. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

Delete a specified name property on a source config template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}/property/{name}
```

Table 513. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template
name	string	true	The name of the property

Get a list of source configs that use the specified source config template

Request

```
GET https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}/sourceConfigs?{parameters}
```

Table 514. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of source config templates to return per page.
pageNumber	string	false	The page of source config templates that should be returned.
orderField	string	false	The field to use when sorting the source config templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the source config templates. This would either be 'asc' or 'desc'.

Table 515. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

Get information about a source config template

Return a JSON representation of a source config template

Request

```
GET https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}
```

Table 516. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

Get information about all source config templates

Return a JSON representation of all source config templates.

Request

```
GET https://{hostname}:{port}
  /build/rest2/sourceTemplates?{parameters}
```

Table 517. Request parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	The number of source config templates to return per page.
pageNumber	int	false	The page of source config templates that should be returned.
orderField	string	false	The field to use when sorting the source config templates. This would either be 'name' or 'id'.
sortType	string	false	The order to use when sorting the source config templates. This would either be 'asc' or 'desc'.
filterValue_name	string	false	The source config template name pattern to search for.

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The source config template description",
  "id": "The ID of the source config template",
  "location": "The URL that can be used to get the JSON
  representation of the source config template",
  "name": "The name of thesource config template",
  "plugin": "The Plug-in ID of Source Plug-in",
  "plugin-properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line",
    "default-value": "The default value of the property",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. Leave blank to use the name as
  the label",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required.",
    "runtime": "Whether or not the value is entered at the
  runtime of a build. When true, the property will appear by
  the Build button. When false, the property will appear in
  the Configuration tab",
    "saveToBuildConfig": "Whether or not to save the
  property to build configurations. This is only used with
  projects, processes, and their respective templates.",
    "scripted": "Whether or not the value type of the
```

```
  property is scripted",
    "value": "The value of the property. This is only used
  for properties on non-template objects (project, but not
  project template, etc."
  }],
  "work-dir-script": "The id of working directory script"
}
```

Get properties on a source template

Request

```
GET https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}/properties?{parameters}
```

Table 518. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of source templates properties to return per page.
pageNumber	string	false	The page of source templates properties that should be returned.

Table 519. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

Update a source config template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}
```

Table 520. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The source config template description",
  "name": "The name of the source config template",
  "plugin-properties": [{
    "name": "The name of the plugin property",
    "value": "The value of the plugin property"
  }],
  "properties": [{
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line if property display-type is Multi-
  Select or Select",
    "allowedValues": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line if property display-type is Multi-
  Select or Select",
    "allowedValuesScript": "The beanshell script to
  determine the allowed values to be selected from if
  property display-type is Multi-Select or Select",
    "default-value": "The default value of this property.",
    "defaultValueCheckbox": "The default value for the
  property when the property's display-type is Checkbox",
    "defaultValueText": "The default value for the
  property when the property's display-type is Text Area",
    "defaultValueTextArea": "The default value of the
  property when the property's display-type is Text Area",
    "defaultValues": "The default value of the property.",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin)",
    "job": "The id of the job when the property is a Job
  Execution property",
    "label": "A label for this property shown when
  prompting users for value. (Leave blank to use the name as
  the Label)",
    "name": "The name of the property",
    "required": "Whether or not the value of the property
  is required when creating a source config using this
  template.",
    "scripted": "Whether or not the value type of the
  property is scripted",
    "valueType": "The value type of the property
  (scripted, defined, jobExecution)"
  }],
  "work-dir-script": "The id of working directory script"
}
```

Update properties on a source config template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/sourceTemplates/{id}/properties
```

Table 521. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a source config. This can be the name or ID of the source config template

stage resource

/stage POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/stage
```

/stage/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/stage/{id}
```

Table 522. URL parameters

Parameter	Type	Required	Description
id	string	true	

/stage/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/stage/{id}
```

Table 523. URL parameters

Parameter	Type	Required	Description
id	string	true	

/stage/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/stage/{id}
```

Table 524. URL parameters

Parameter	Type	Required	Description
id	string	true	

/stage/{id}/{path:.+} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/stage/{id}/{path:.+}
```

Table 525. URL parameters

Parameter	Type	Required	Description
path	string	true	
id	string	true	

/stage/{id}/{path:.+} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/stage/{id}/{path:.+}
```

Table 526. URL parameters

Parameter	Type	Required	Description
path	string	true	
id	string	true	

statuses resource

/statuses GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/statuses
```

/statuses POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/statuses
```

/statuses/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/statuses/{id}
```

Table 527. URL parameters

Parameter	Type	Required	Description
id	string	true	

/statuses/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/statuses/{id}
```

Table 528. URL parameters

Parameter	Type	Required	Description
id	string	true	

/statuses/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/statuses/{id}
```

Table 529. URL parameters

Parameter	Type	Required	Description
id	string	true	

Get the statuses which are not archived.

Request

```
GET https://{hostname}:{port}
  /build/rest2/statuses/nonArchived
```

stepPreConditionScripts resource

/stepPreConditionScripts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/stepPreConditionScripts
```

/stepPreConditionScripts POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/stepPreConditionScripts
```

/stepPreConditionScripts/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/stepPreConditionScripts/{id}
```

Table 530. URL parameters

Parameter	Type	Required	Description
id	string	true	

/stepPreConditionScripts/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/stepPreConditionScripts/{id}
```

Table 531. URL parameters

Parameter	Type	Required	Description
id	string	true	

/stepPreConditionScripts/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/stepPreConditionScripts/{id}
```

Table 532. URL parameters

Parameter	Type	Required	Description
id	string	true	

steps resource

/steps GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/steps
```

/steps/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/steps/{id}
```

Table 533. URL parameters

Parameter	Type	Required	Description
id	string	true	

system resource

/system/properties GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/system/properties?{parameters}
```

Table 534. Request parameters

Parameter	Type	Required	Description
pageNumber	int	false	The page of system properties that should be returned.
rowsPerPage	int	false	The number of system properties to return per page.

Create a property in system propertes

Request

```
POST https://{hostname}:{port}
  /build/rest2/system/properties
```

Delete a specified name property in system propertes

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/system/property/{name}
```

Table 535. URL parameters

Parameter	Type	Required	Description
name	string	true	The name of the property

Update properties in system propertes

Request

```
PUT https://{hostname}:{port}
  /build/rest2/system/properties
```

team resource

team GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team
```

team POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/team
```

team/{team} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}
```

Table 536. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/team/{team}
```

Table 537. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/team/{team}
```

Table 538. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/roleMappings GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/roleMappings
```

Table 539. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/roleMappings POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/team/{team}/roleMappings
```

Table 540. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/roleMappings DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/team/{team}/roleMappings
```

Table 541. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/roleMappings/{role}/validMembers GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/roleMappings/{role}/validMembers
```

Table 542. URL parameters

Parameter	Type	Required	Description
role	string	true	
team	string	true	

team/{team}/resourceMappings GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/resourceMappings
```

Table 543. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/resourceMappings POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/team/{team}/resourceMappings
```

Table 544. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/resourceMappings DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/team/{team}/resourceMappings
```

Table 545. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/resourceMappings/{type} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/resourceMappings/{type}?{parameters}
```

Table 546. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	
pageNumber	string	false	

Table 547. URL parameters

Parameter	Type	Required	Description
team	string	true	
type	string	true	

team/{team}/batchResourceMappings PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/team/{team}/batchResourceMappings
```

Table 548. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/batchResourceMappings DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/team/{team}/batchResourceMappings
```

Table 549. URL parameters

Parameter	Type	Required	Description
team	string	true	

team/{team}/unmappedResources/{type} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/unmappedResources/{type}?{parameters}
```

Table 550. Request parameters

Parameter	Type	Required	Description
name	string	false	

Table 551. URL parameters

Parameter	Type	Required	Description
team	string	true	
type	string	true	

team/{team}/unmappedResources/{type}/{role} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/team/{team}/unmappedResources/{type}/{role}?{parameters}
```

Table 552. Request parameters

Parameter	Type	Required	Description
name	string	false	

Table 553. URL parameters

Parameter	Type	Required	Description
role	string	true	
team	string	true	
type	string	true	

templates resource

Resources for working with project templates.

Create a new property by copy on a project template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/templates/{id}/copyproperty
```

Table 554. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

Create a project template

Request

```
POST https://{hostname}:{port}
  /build/rest2/templates
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the project template is
  active.",
  "build-pre-process": [{
    "name": "The name of the Build Pre-Processing property
  (quiet-period, merge-period, label-values, stamp-values,
  agent-pool, shouldCleanup, pre-condition-script, agent-
  pool-prop).",
    "value": "The value of the Build Pre-Processing
  property"
  }],
  "description": "The project template description",
  "name": "The name of the project template",
  "properties": [{
    "allowed-values": "The values users are allowed to
```

```
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "default-value": "The default value of this property.",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "label": "A label for this property shown when
  prompting users for value. (Leave blank to use the Name as
  the Label)",
    "name": "The name of the property",
    "required": "whether or not the value of the property
  is required when creating a project using this template.",
    "scripted": "whether or not the value type of the
  property is scripted"
  }],
  "teams": [{
    "resourceRoleId": "The id of the resource role",
    "teamId": "The id of the team"
  }]
}
```

Delete a project template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/templates/{id}
```

Table 555. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

Delete a specified name property on a project template

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/templates/{id}/property/{name}
```

Table 556. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

Table 556. URL parameters (continued)

Parameter	Type	Required	Description
name	string	true	The name of the property

Get a list of projects that use the specified project template

Request

```
GET https://{hostname}:{port}
  /build/rest2/templates/{id}/projects?{parameters}
```

Table 557. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of projects to return per page.
pageNumber	string	false	The page of projects that should be returned.
orderField	string	false	The field to use when sorting the projects. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the projects. This would either be 'asc' or 'desc'.

Table 558. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

Get information about a project template

Return a JSON representation of a project template

Request

```
GET https://{hostname}:{port}
  /build/rest2/templates/{id}
```

Table 559. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the project template is
  active.",
  "build-pre-process": [{
    "name": "The name of the Build Pre-Processing property
  (quiet-period, merge-period, label-values, stamp-values,
  agent-pool, shouldCleanup, pre-condition-script, agent-
  pool-prop).",
    "value": "The value of the Build Pre-Processing
  property"
  }],
  "description": "The project template description",
  "id": "The ID of the project template",
  "location": "The URL that can be used to get the JSON
  representation of the project template",
  "name": "The name of the project template",
  "properties": [{
    "agentPool": "The agent pool of the property",
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValues": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValuesScript": "The beanshell script to
  determine the allowed values to be selected from, if
  property display-type is Multi-Select or Select",
    "default-value": "The default value of this property.",
    "defaultAgentPool": "The default agent pool of the
  property",
    "defaultValueCheckbox": "The default value for the
  property when the property's display-type is Checkbox",
    "defaultValueText": "The default value for the
  property when the property's display-type is Text Area",
    "defaultValueTextArea": "The default value of the
  property when the property's display-type is Text Area",
    "defaultValues": "The default value of the property.",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "isNew": "whether or not this is a new property",
    "job": "The id of the job",
    "label": "A label for this property shown when
  prompting users for value. (Leave blank to use the name as
```

```
  the Label)",
    "name": "The name of the property",
    "oldName": "The name of the property if this is an
  existing property",
    "pluginType": "The plugin type of the property",
    "required": "whether or not the value of the property
  is required when creating a project using this template.",
    "scripted": "whether or not the value type of the
  property is scripted",
    "valueType": "The value type of the property
  (scripted, defined, jobExecution)"
  }],
  "resourceId": "The security resource ID of the project
  template"
}
```

Get information about all project templates

Return a JSON representation of all project templates.

Request

```
GET https://{hostname}:{port}
  /build/rest2/templates?{parameters}
```

Table 560. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of project templates to return per page.
pageNumber	string	false	The page of project templates that should be returned.
orderField	string	false	The field to use when sorting the project templates. This would either be 'name' or 'description'.
sortType	string	false	The order to use when sorting the project templates. This would either be 'asc' or 'desc'.

This command takes a JSON request string or file. Use the following template for the request:

```
[{
  "active": "Whether or not the project template is
  active.",
  "build-pre-process": [{
    "name": "The name of the Build Pre-Processing property
  (quiet-period, merge-period, label-values, stamp-values,
  agent-pool, shouldCleanup, pre-condition-script, agent-
  pool-prop).",
    "value": "The value of the Build Pre-Processing
  property"
  }],
  "description": "The project template description",
```

```
  "id": "The ID of the project template",
  "location": "The URL that can be used to get the JSON
  representation of the project template",
  "name": "The name of the project template",
  "properties": [{
    "agentPool": "The agent pool of the property",
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValues": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValuesScript": "The beanshell script to
  determine the allowed values to be selected from, if
  property display-type is Multi-Select or Select",
    "default-value": "The default value of this property.",
    "defaultAgentPool": "The default agent pool of the
  property",
    "defaultValueCheckbox": "The default value for the
  property when the property's display-type is Checkbox",
    "defaultValueText": "The default value for the
  property when the property's display-type is Text Area",
    "defaultValueTextArea": "The default value of the
  property when the property's display-type is Text Area",
    "defaultValues": "The default value of the property.",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "isNew": "whether or not this is a new property",
    "job": "The id of the job",
    "label": "A label for this property shown when
  prompting users for value. (Leave blank to use the name as
  the Label)",
    "name": "The name of the property",
    "oldName": "The name of the property if this is an
  existing property",
    "pluginType": "The plugin type of the property",
    "required": "whether or not the value of the property
  is required when creating a project using this template.",
    "scripted": "whether or not the value type of the
  property is scripted",
    "valueType": "The value type of the property
  (scripted, defined, jobExecution)"
  }],
  "resourceId": "The security resource ID of the project
  template"
}]
```

Get properties on a project template

Request

```
GET https://{hostname}:{port}
  /build/rest2/templates/{id}/properties?{parameters}
```

Table 561. Request parameters

Parameter	Type	Required	Description
rowsPerPage	string	false	The number of project template properties to return per page.
pageNumber	string	false	The page of project template properties that should be returned.

Table 562. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

Update a project template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/templates/{id}
```

Table 563. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "Whether or not the project template is
  active.",
  "build-pre-process": [{
    "name": "The name of the Build Pre-Processing property
  (quiet-period, merge-period, label-values, stamp-values,
  agent-pool, shouldCleanup, pre-condition-script, agent-
  pool-prop).",
    "value": "The value of the Build Pre-Processing
  property"
  }],
  "description": "The project template description",
  "name": "The name of the project template",
  "properties": [{
    "agentPool": "The agent pool of the property",
    "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
```

```
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValues": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
    "allowedValuesScript": "The beanshell script to
  determine the allowed values to be selected from, if
  property display-type is Multi-Select or Select",
    "default-value": "The default value of this property.",
    "defaultAgentPool": "The default agent pool of the
  property",
    "defaultValueCheckbox": "The default value for the
  property when the property's display-type is Checkbox",
    "defaultValueText": "The default value for the
  property when the property's display-type is Text Area",
    "defaultValueTextArea": "The default value of the
  property when the property's display-type is Text Area",
    "defaultValues": "The default value of the property.",
    "description": "The description of the property",
    "display-type": "The display type of the property
  (Text, Select, Checkbox, Multi-Select, Text Area, Text
  (secure), Integration Plugin, Agent Pool)",
    "isNew": "whether or not this is a new property",
    "job": "The id of the job",
    "label": "A label for this property shown when
  prompting users for value. (Leave blank to use the name as
  the Label)",
    "name": "The name of the property",
    "oldName": "The name of the property if this is an
  existing property",
    "pluginType": "The plugin type of the property",
    "required": "whether or not the value of the property
  is required when creating a project using this template.",
    "scripted": "whether or not the value type of the
  property is scripted",
    "valueType": "The value type of the property
  (scripted, defined, jobExecution)"
  }]
}
```

Update properties on a project template

Request

```
PUT https://{hostname}:{port}
  /build/rest2/templates/{id}/properties
```

Table 564. URL parameters

Parameter	Type	Required	Description
id	string	true	The identifier to use when looking up a project template. This can be the name or ID of the project template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agentPool": "The agent pool of the property",
  "allowed-values": "The values users are allowed to
  select for this property. Separate each value by entering
  it on its own line, if property display-type is Multi-
  Select or Select",
  "allowedValues": "The values users are allowed to select
  for this property. Separate each value by entering it on
  its own line, if property display-type is Multi-Select or
  Select",
  "allowedValuesScript": "The beanshell script to
  determine the allowed values to be selected from, if
  property display-type is Multi-Select or Select",
  "default-value": "The default value of this property.",
  "defaultAgentPool": "The default agent pool of the
  property",
  "defaultValueCheckbox": "The default value for the
  property when the property's display-type is Checkbox",
  "defaultValueText": "The default value for the property
  when the property's display-type is Text Area",
  "defaultValueTextArea": "The default value of the
  property when the property's display-type is Text Area",
  "defaultValues": "The default value of the property.",
  "description": "The description of the property",
  "display-type": "The display type of the property (Text,
  Select, Checkbox, Multi-Select, Text Area, Text (secure),
  Integration Plugin, Agent Pool)",
  "isNew": "whether or not this is a new property",
  "job": "The id of the job",
  "label": "A label for this property shown when prompting
  users for value. (Leave blank to use the name as the
  Label)",
  "name": "The name of the property",
  "oldName": "The name of the property if this is an
  existing property",
  "pluginType": "The plugin type of the property",
  "required": "whether or not the value of the property is
  required when creating a project using this template.",
  "scripted": "whether or not the value type of the
  property is scripted",
  "valueType": "The value type of the property (scripted,
  defined, jobExecution)"
}
```

trends resource

/trends/tests/{name}/{curr}/since/{since} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/tests/{name}/{curr}/since/{since}
```

Table 565. URL parameters

Parameter	Type	Required	Description
name	string	true	
curr	string	true	
since	string	true	

/trends/coverages/{name}/{curr} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/coverages/{name}/{curr}
```

Table 566. URL parameters

Parameter	Type	Required	Description
name	string	true	
curr	string	true	

/trends/coverages/{name}/{curr}/since/{since} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/coverages/{name}/{curr}/since/{since}
```

Table 567. URL parameters

Parameter	Type	Required	Description
name	string	true	
curr	string	true	
since	string	true	

/trends/analytics/{name}/{curr}/{limit} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/analytics/{name}/{curr}/{limit}
```

Table 568. URL parameters

Parameter	Type	Required	Description
name	string	true	
limit	int	true	
curr	string	true	

/trends/analytics/{name}/{curr}/since/counts/{since} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/analytics/{name}/{curr}/since/counts/{since}
```

Table 569. URL parameters

Parameter	Type	Required	Description
name	string	true	
curr	string	true	
since	string	true	

/trends/analytics/{name}/{curr}/since/severities/{since} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/trends/analytics/{name}/{curr}/since/severities/{since}
```

Table 570. URL parameters

Parameter	Type	Required	Description
name	string	true	
curr	string	true	
since	string	true	

user resource

user GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/user?{parameters}
```

Table 571. Request parameters

Parameter	Type	Required	Description
name	string	false	

user POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/user
```

user/{user} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/user/{user}
```

Table 572. URL parameters

Parameter	Type	Required	Description
user	string	true	

user/{user} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/user/{user}
```

Table 573. URL parameters

Parameter	Type	Required	Description
user	string	true	

user/{user} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/user/{user}
```

Table 574. URL parameters

Parameter	Type	Required	Description
user	string	true	

user/{user}/setPassword PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/user/{user}/setPassword
```

Table 575. URL parameters

Parameter	Type	Required	Description
user	string	true	

user/{user}/unlock PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/user/{user}/unlock
```

Table 576. URL parameters

Parameter	Type	Required	Description
user	string	true	

users resource

users GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/users
```

users/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/users/{id}
```

Table 577. URL parameters

Parameter	Type	Required	Description
id	string	true	

users/{id}activate PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/users/{id}activate
```

Table 578. URL parameters

Parameter	Type	Required	Description
id	string	true	

users/{id}inactivate PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/users/{id}inactivate
```

Table 579. URL parameters

Parameter	Type	Required	Description
id	string	true	

Create a user

Request

```
POST https://{hostname}:{port}
  /build/rest2/users
```

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "actualName": "The full name of the user.",
  "emailAddress": "The email address to use for
```

```
  notifications.",
  "imId": "The instant messaging ID for notifications.",
  "locale": "The locale of this user.",
  "name": "The user name to use to log into the product.
  Required.",
  "numDashRows": "The number of recent items to display on
  the dashboard.",
  "password": "The password for this user. Required.",
  "passwordConfirm": "Confirm the new password. Required.",
  "realmId": "The ID of Authentication Realm you wish to
  create the user in.",
  "receiveNotifications": "Receive notifications from the
  server. Clear this check box to unsubscribe all emails
  from the server.",
  "shortDateFormat": "The date and time format to use on
  the dashboard. The default is based upon the locale of the
  server.",
  "timeZone": "The time zone of the user.",
  "use24Clock": "The boolean value to indicate whether to
  use 24-Hour Clock",
  "userGroup": "The JSONArray of group ID."
}
```

Get information about a user

Return a JSON representation of the user

Request

```
GET https://{hostname}:{port}
  /build/rest2/users/{id}
```

Table 580. URL parameters

Parameter	Type	Required	Description
id	string	true	

Update a user

Request

```
POST https://{hostname}:{port}
  /build/rest2/users/{id}/update
```

Table 581. URL parameters

Parameter	Type	Required	Description
id	string	true	

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "actualName": "The full name of the user.",
  "emailAddress": "The email address to use for
  notifications.",
  "imId": "The instant messaging ID for notifications.",
  "locale": "The locale of this user.",
  "name": "The user name to use to log into the product.",
  "numDashRows": "The number of recent items to display on
  the dashboard.",
  "password": "The password for this user.",
  "receiveNotifications": "The boolean value indicate
  whether to receive notifications from the server.",
  "shortDateFormat": "The date and time format to use on
  the dashboard. The default is based upon the locale of the
  server.",
  "timeZone": "The time zone of the user.",
  "use24Clock": "The boolean value indicate whether to use
  24-Hour Clock",
  "userGroup": "The JSONArray of group ID."
}
```

workDirScripts resource

/workDirScripts GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workDirScripts
```

/workDirScripts POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workDirScripts
```

/workDirScripts/{type} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workDirScripts/{type}
```

Table 582. URL parameters

Parameter	Type	Required	Description
type	string	true	

/workDirScripts/{id} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workDirScripts/{id}
```

Table 583. URL parameters

Parameter	Type	Required	Description
id	string	true	

/workDirScripts/{id} PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/workDirScripts/{id}
```

Table 584. URL parameters

Parameter	Type	Required	Description
id	string	true	

/workDirScripts/{id} DELETE

Request

```
DELETE https://{hostname}:{port}
  /build/rest2/workDirScripts/{id}
```

Table 585. URL parameters

Parameter	Type	Required	Description
id	string	true	

workflow/{workflowId/dependencies resource

/workflow/{workflowId}/dependencies/ PUT

Request

```
PUT https://{hostname}:{port}
  /build/rest2/workflow/{workflowId}/dependencies/
```

Table 586. URL parameters

Parameter	Type	Required	Description
workflowId	long	true	

workflow/massdependencytree resource

/workflow/massdependencytreenodes GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreenodes
```

/workflow/massdependencytreenodes/{nodeId} GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreenodes/{nodeId}
```

Table 587. URL parameters

Parameter	Type	Required	Description
nodeId	string	true	

/workflow/massdependencytreemoveDependencyAjax POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreemoveDependencyAjax
```

/workflow/massdependencytreecopyDependencyAjax POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreecopyDependencyAjax
```

/workflow/massdependencytreecreateWorkflowDependencyAjax POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreecreateWorkflowDependencyAjax
```

/workflow/massdependencytreecreateCodestationDependencyAjax POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreecreateCodestationDependencyAjax
```

/workflow/massdependencytreedeleteDependencyAjax POST

Request

```
POST https://{hostname}:{port}
  /build/rest2/workflow/massdependencytreedeleteDependencyAjax
```

workflowStatuses resource

/workflowStatuses GET

Request

```
GET https://{hostname}:{port}
  /build/rest2/workflowStatuses
```

conrefs

```
-username admin -password admin
  -weburl http://myserver.example.com:8080
```

```
-k -u jsmith:passwd
```

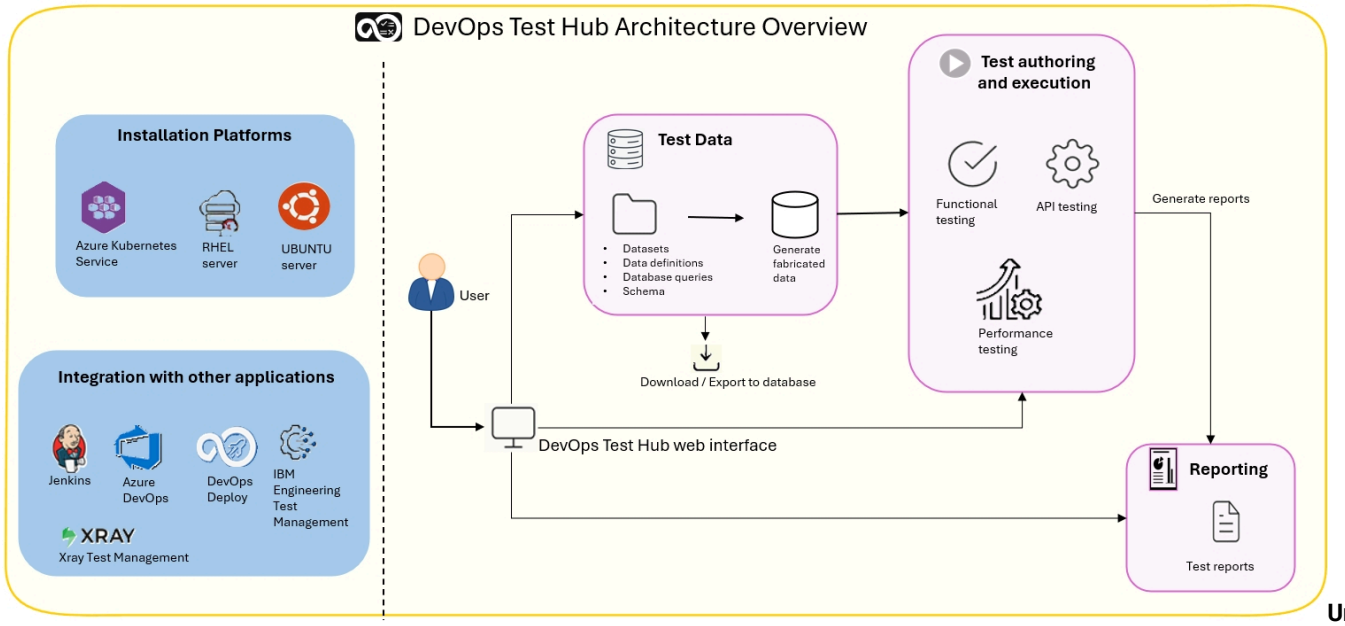
```
http://myserver.example.com:8080
```

Test

The Test capability in DevOps Loop is powered by the DevOps Test application. Test centralizes test data, environments, test runs, and reports in a single web based interface within the DevOps Loop. It streamlines test management, execution, and tracking, providing greater efficiency and visibility to ensure quality throughout the loop.

Introduction to DevOps Test Hub

Test is a web-based continuous testing platform that consolidates test data, environments, test runs, and reports. It is built on cloud-native technologies and Docker, and it supports API, functional, and performance tests. Test offers scalability across cloud and on-premises systems with Kubernetes and IBM® Red Hat OpenShift support.

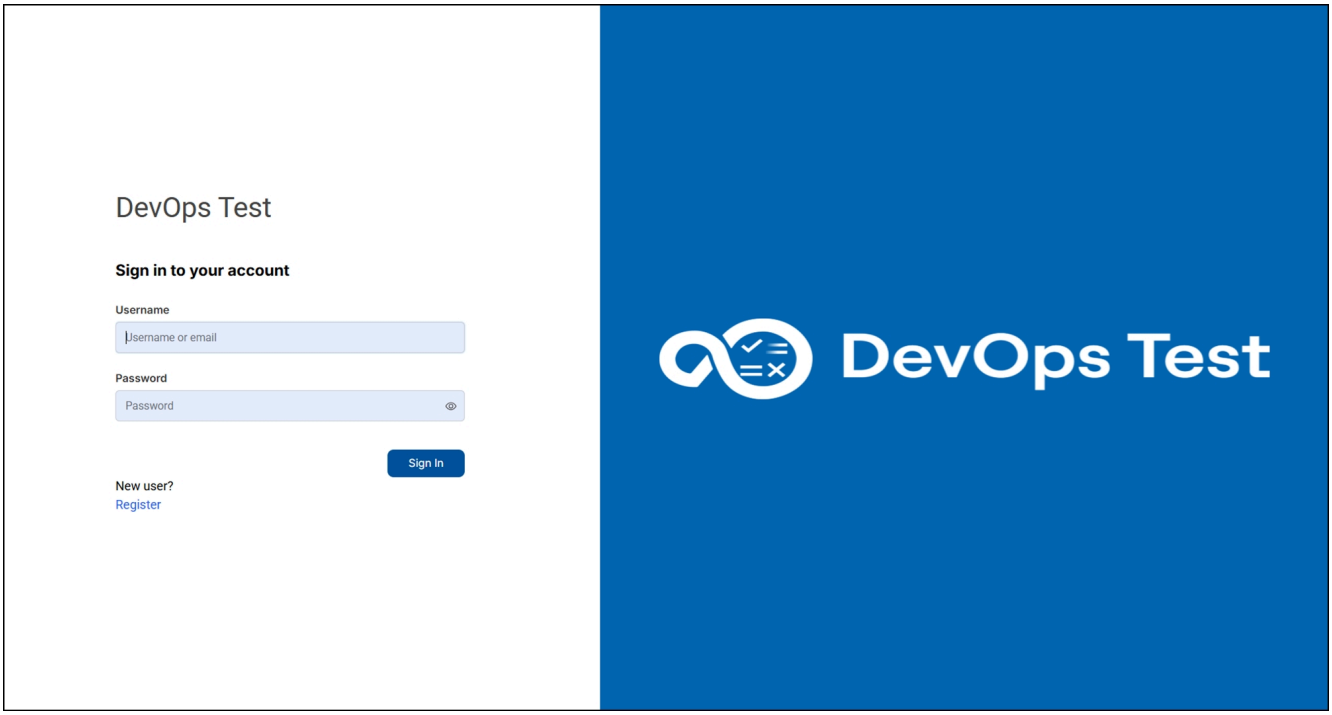


Unified test dashboard

You can use the intuitive dashboard to create, manage, and track test activities. By using the dashboard, you can efficiently manage test data and collaborate with your team on test requirements.

Login page

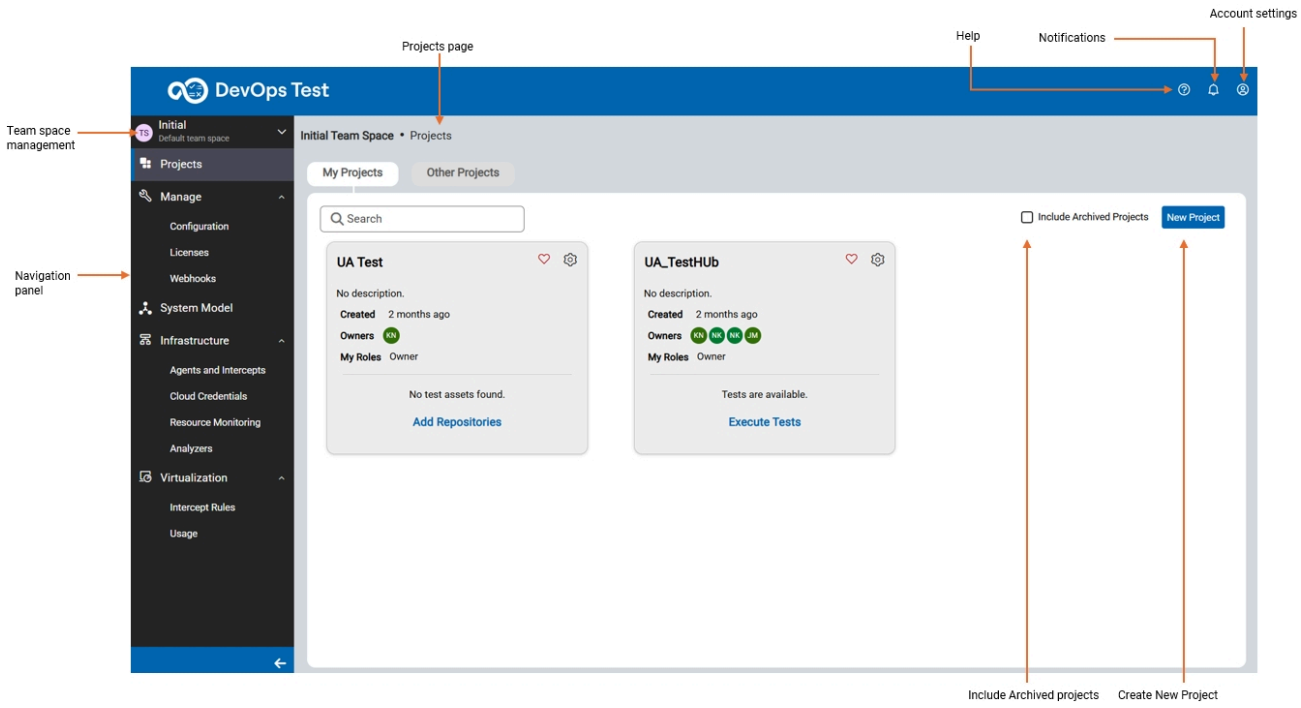
The login page is the primary access point to the platform. Only registered users can access project data and testing environments. If you are a new user, you must complete the registration process before you can log in.



Note: A session timeout prompt is displayed if you are inactive for 30 minutes. Before the session ends, a notification appears, prompting you to **continue the session** or **log out**. If no action is taken, the session is automatically terminated.

Home page

The home page lists your projects and other projects, which makes it easy for you to manage different projects within an organization. You can request access to join existing projects, create a new project, and view archived projects.



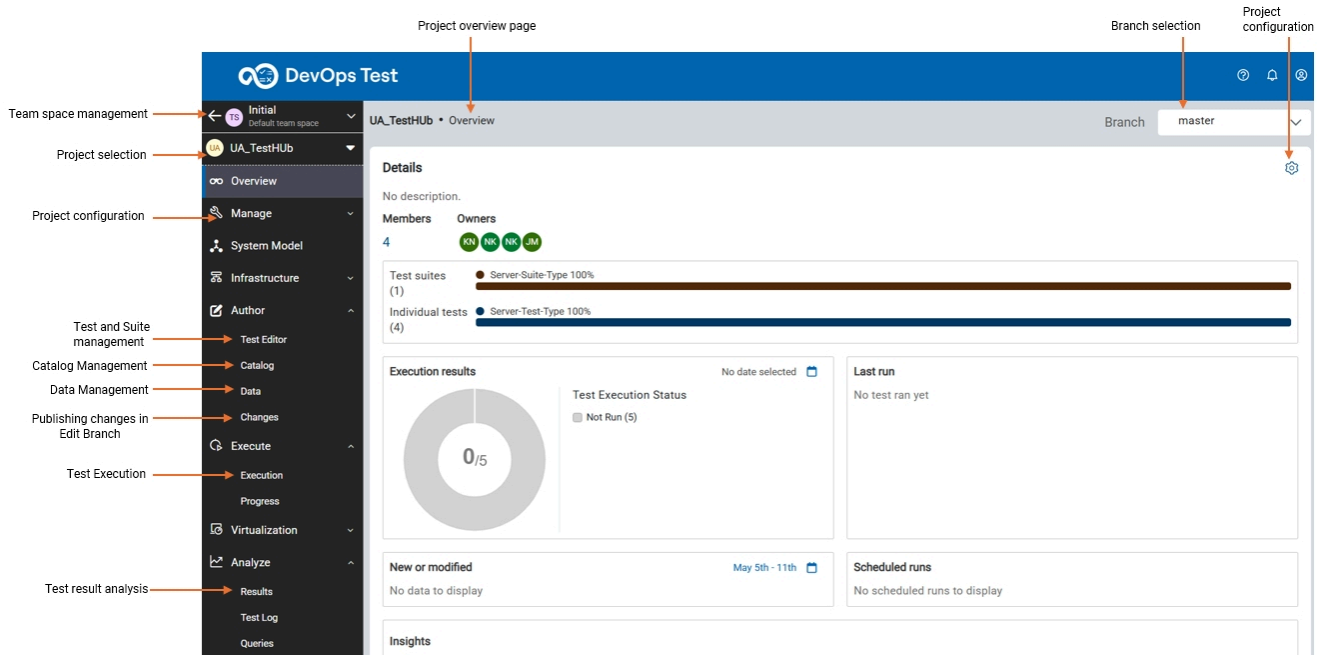
Navigation panel

The navigation panel in Test Hub provides quick access to sections such as **Projects**, **Manage**, **System Model**, **Infrastructure**, and **Virtualization**. You can easily navigate your projects, configure settings, define system models, monitor infrastructure, and manage virtualization. Each section of the dashboard is designed to support you with a specific function, making your testing workflow more efficient and intuitive.

After you select a project, the panel displays the **Author**, **Execute**, and **Analyze** sections for test creation, data management, test execution, and reporting activities.

Overview page

You can manage and monitor all testing activities in one place on the project **Overview** page.



Logout

When you want to end your session with Test, you can log out by clicking the profile icon , and selecting **Logout**.

Click here to watch the visual walkthrough.

Related information

Capabilities on page 703

Installation

User tasks on page 703

Capabilities

Test unifies test data, test environments, execution, and reporting in a web-based platform designed for both testers and non-testers. You can efficiently manage test data, collaborate seamlessly with your team, and integrate your testing workflows with DevOps pipelines for streamlined delivery and execution.

Test data authoring and management

You can create, edit, and manage test data directly in Test. With support for concurrent multi-user editing, and teams can work on test datasets at the same time without causing conflicts or overwriting each other's changes. The built-in data generator helps to generate fabricated data for different testing scenarios.

Role-based access and security

Testoffers robust role-based access control, where project owners can assign permissions according to defined user roles. This ensures secure test data management and controlled access to sensitive assets such as credentials.

Report analysis and resource monitoring

Test Hub includes advanced reporting capabilities by offering insights into test execution, system performance, and resource utilization. It consolidates unified test results, helping teams make data-driven decisions to improve software quality.

Connected performance testing agents

You can link existing performance testing agents to Test to execute schedules and Accelerated Functional Testing (AFT) suites by using your current infrastructure for seamless integration.

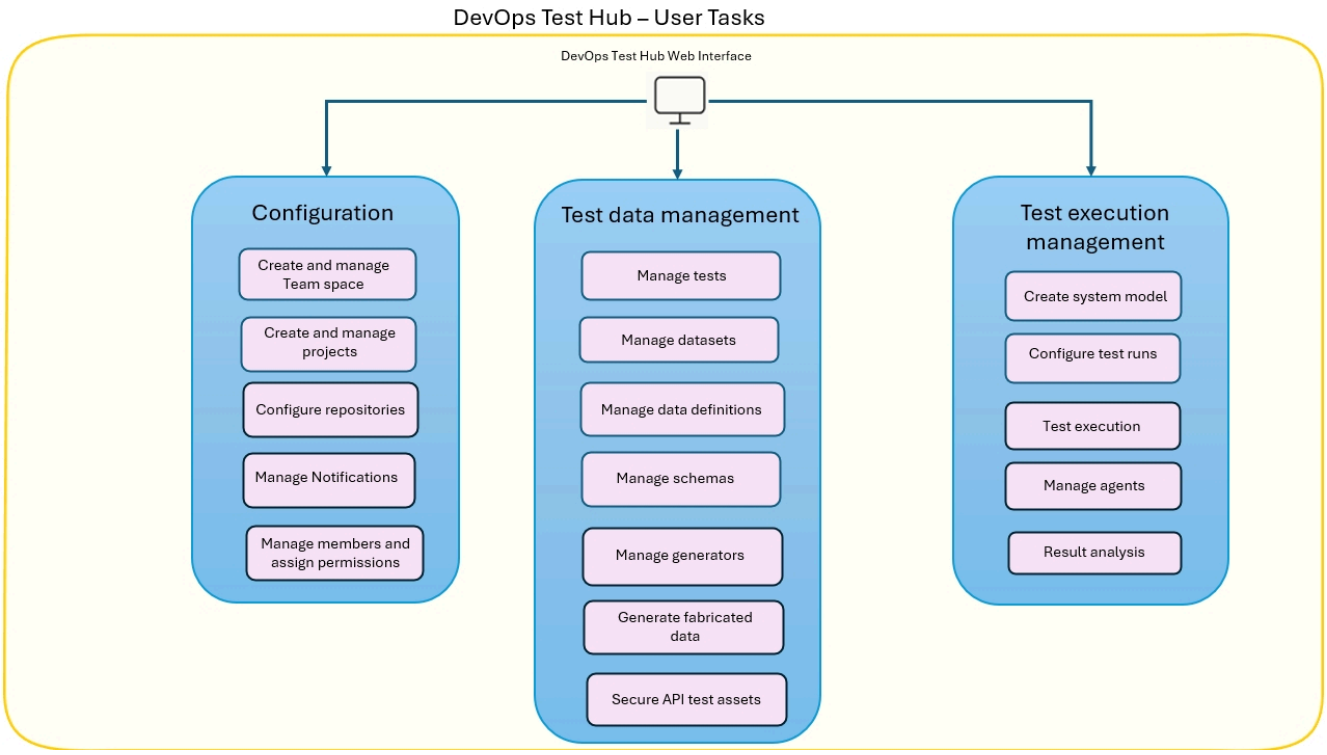
Integration with DevOps tools

You can integrate Test with popular DevOps tools like Jenkins, Azure DevOps, DevOps Deploy, Jira, and GitHub. These integrations help you automate test execution within your CI/CD pipelines and streamline defect-tracking workflows.

User tasks

Test offers a range of user tasks designed to simplify collaboration and optimize testing workflows. You can efficiently create, manage, and analyze test data to ensure a smooth and effective testing process.

You can perform the following user tasks in Test to support different stages of the testing lifecycle.



Member access and permissions

You can manage access in Test by assigning roles and permissions to control who can view, edit, or execute test assets within a project. With effective access management, you can protect your project data while making collaboration easier and more secure for your entire team.

For more information on configuring your Test account, see the Configuration of DevOps Test on page 730 section in the Administrator Guide on page 730.

For more information on roles, access, and permissions, see Managing members and their roles in a team space on page 1721 and Managing access to the server projects on page 1735 in the Test Manager Guide on page 1713.

Compatibility and supported assets

You can find information about the environments, file formats, and asset types that are compatible with Test . You can use this information to verify if your assets and environments are supported before creating or importing tests into Test.

Related information

Supported versions of assets and resources on page 705

Supported suites and tests on page 706

Supported virtual services on page 707

Supported versions of assets and resources

You can find information about the versions of assets, resources, and agents that are supported in Test . You can also find information about the versions of the desktop clients that are supported.

The following table lists the versions of assets or resources created in the desktop clients along with the agents, or proxies that are supported in Test:

Resource	Supported	Not supported
Assets or resources created in HCL DevOps Test UI (Test UI)	Assets or resources created in Test UI that is of a version earlier than or the same as Test.	Assets or resources created in any later version of Test UI that is later than the version of Test.
Assets or resources created in HCL DevOps Test Integrations and APIs (Test Integrations and APIs)	Assets or resources created in Test Integrations and APIs that is of a version earlier than or the same as Test.	Assets or resources created in any later version of Test Integrations and APIs that is later than the version of Test.
Assets or resources created in HCL DevOps Test Performance (Test Performance)	Assets or resources created in Test Performance that is of a version earlier than or the same as Test.	Assets or resources created in any later version of Test Performance that is later than the version of Test.
HCL DevOps Test Performance Agent (Test Performance Agent)	Test Performance Agent, which is the same version as Test.	Before you install the agents that are of the current version, see Management of performance agents on page 1072.
HCL DevOps Test Integrations and APIs Agent (Test Integrations and APIs Agent)	Test Integrations and APIs Agent, which is the same version as Test.	Before you install the agents that are of the current version, see Management of API agents on page 1080.
HTTP proxies	Installation of the proxy component from Test Virtualization Control Panel, which is the same version as Test.	Before you use HTTP proxies, see Setting up intercepts on page 1457.

If there is a mismatch in the versions of the assets, resources, or agents with the version of Test that you are using, any of the following events can occur:

- Warnings or Errors are displayed when you add a repository to a project in your team space.
- Warnings or Errors are displayed when you open a project that you migrated from a previous version of Test.
- Warnings or Errors are displayed when you add or manually refresh a repository to which you added assets or resources that were created on a desktop client with a version previous to or later than the version of Test that you want to use.
- Test runs that you start or schedule on an agent fail to run.
- You cannot view the proxies that are registered with Test on the **Agents and Intercepts** page.

Supported suites and tests

You can find information about the types of Test Suites and tests that you can configure for a run in a project on Test .

The following table lists the Test Suites and test types that are supported and those tests that are not supported to run on Test:

Supported Suites	Supported tests	Unsupported tests
• Accelerated Functional Testing (AFT) Suites • Compound Tests that contain any of the following types of tests: ◦ Web UI tests ◦ HTML tests ◦ Mobile tests	• Individual Web UI test • Individual Windows test • Individual Mobile test	• Traditional HTML tests • Traditional non-HTML tests
• API Suites that include the following Test Suites: ◦ Test Suites that contain scenarios with references to local stubs. ◦ Test Suites that contain stubs that virtualize services run in any Kubernetes cluster or in the Kubernetes cluster that hosts Test. • Virtual services or Stubs	API tests	Suites or tests with subscribe actions that operate in the watch mode.
• Compound Tests that contain performance tests • Rate Schedules • VU Schedules	Individual Performance test	• A single test script of any test extension. • Tests that belong to 32-bit test extensions and SOA Quality included in VU Schedule, Rate Schedule, and Compound Test. • Citrix, SAP, or Siebel as an individual test or as part of a Compound Test.
Postman resources		
JMeter tests		

Supported Suites	Supported tests	Unsupported tests
JUnit tests		
Selenium tests		

Note: You must create Compound Tests for the same type of tests. For example, you can create a Compound Test that has different mobile tests that use different mobile testing platforms.

Restriction: You cannot merge folders that contain different test projects in the same project directory. For example, you cannot create a project directory that contains a Performance test project and an API Suite project. You must create separate project directories for each test type. For example, you must create the Performance test in the project directory as `/MyPerfProject/` and the API Suite in the project directory as `/MyAPIProject`, before you commit the projects to the remote repository.

Related information

Test run configurations on page 1134

Supported virtual services

You can find information about the virtual services that are supported on Test .

You can start a running instance of the virtual services from Test to run on the following locations:

- A registered API agent.
- The default Kubernetes cluster of Test.
- A remote Kubernetes cluster.

You can run all virtual services on registered API agents except the virtual services that use CICS transport, DB2/Z transport, or Istio.

The default Kubernetes cluster or a remote Kubernetes cluster supports running of the following types of virtual services:

- Virtual services that utilize the WebSphere® MQ transport.
- Virtual services that utilize the HTTP transport.

Related information

Management of virtualized services on page 1453

Accessibility features

Accessibility features help users who have physical disabilities, such as visual and, hearing impairment, or limited mobility, to use the software products successfully.

Accessibility compliance

The product documentation is published by using Oxygen XML WebHelp Responsive. To understand the accessibility compliance status for Oxygen XML WebHelp Responsive, refer to WebHelp Responsive VPAT Accessibility Conformance Report.

Accessing UI elements

Test supports navigation in the UI by using different methods such as a mouse, keyboard, or touchpad.

You can use the keyboard keys such as **Tab**, arrow keys such as **UP**, **DOWN**, **LEFT**, and **RIGHT** to navigate to the different pages in the **Navigation** pane or to the different action labels in the right pane on the UI.

Tutorials

Learn how to perform frequently used tasks, including creating and managing Web UI tests and viewing test results through functional reports.

Creation and management of your first Web UI test

In the lessons detailed in this learning module, you can learn how to use DevOps Test to create and manage Web UI tests.

Steps to build and run a Web UI test in Test Hub

In this section, you will learn about the typical steps involved in building and running a Web UI test using DevOps Test. Throughout this tutorial, you will follow these steps to create, enhance, execute, and analyze Web UI tests.

You will perform the following tasks:

1. **Create an application**: Set up the web application that you want to test within the project.
2. **Record and manage Web UI test steps**: Capture user interactions such as clicks, text entry, and selections, and organize them into reusable test steps.
3. **Add variables and validations to tests**: Make your tests reusable and verify that the application behaves as expected.
4. **Run tests and view functional reports**: Run tests and review results to confirm whether they passed or failed.

As a real-time scenario, the tutorial uses an HR management web application to demonstrate recording user interactions such as logging in and managing employee details.

You can apply this workflow to any web application. Just use your own application URL instead of the example shown here and follow the steps outlined in this tutorial.

Prerequisites

Before you begin, ensure that your system is ready to complete the tutorial.

The prerequisites are as follows:

1. Your system must have at least one supported web browser installed, such as Google Chrome or Microsoft Edge.
2. A DevOps Test user account.
3. A project that is configured with a Git repository.

Now, you will learn how to perform the prerequisite tasks in detail, starting with creating a DevOps Test user account.

Create a new user account

In this lesson, you will learn how to create a new user account in the DevOps Test. If LDAP or Active Directory authentication is not configured, you must create a user account to get started.

About this task

To access the DevOps Test and perform Web UI testing tasks, you must have a user account. If you already have an account, you can sign in directly. Otherwise, you have to follow the steps below to create one.

1. Open a web browser and access the DevOps Test URL for your environment.
   For example:
   ```
   https://<host-name>/devops-test
   ```
2. Click **Register**.
3. Enter the following details to complete the registration:
   - **Email address**
   - **Username**
   - **Password**
   - **Confirm password**
4. Click **Register**.

Results

You have successfully created a user account and can now access the DevOps Test.

What to do next

Next, you will create a project. See Create a project on page 710.

Create a project

In this lesson, you will learn how to create a project and associate a Git repository to start recording and managing Web UI tests. A project defines the workspace where your test assets are stored and managed.

About this task

When you create a project, you are automatically assigned the Project Creator and Architect roles for the **Initial Team Space**.

1. Click **New project** to create a new project in the **Initial Team Space**.
2. Enter a name in the **Project** name field. For example, *myproj*
3. (optional) Enter a brief description in the **Description** field.
4. Click **Create**.

   The **myproj** project is created successfully.
5. On the **Configuration** page, under **Repositories**, click **Add Repository** to associate a repository with the project.
6. Enter the complete URL of the Git repository that you want to associate with the project.
   For example,

   ```
   https://github.com/<user>/<repository>.git
   ```
7. Enter the required credentials (username and password) to authenticate access to the repository.

   The Git repository is associated with the project.

Results

The project is created successfully, and the repository is associated with the project.

After completing the prerequisites, you are now ready to proceed with creating an application.

What to do next

You can now begin creating an application. See Lesson 1 on page 710.

Lesson 1: Create an application

In this lesson, you will learn how to create an application within your project to record a Web UI test for an HR management application. An application represents the web application that you want to test.

Before you begin

Ensure that you have completed all the Prerequisite on page 709 tasks.

About this task

An application stores the URL and other related configuration details of the web application that you want to record and manage Web UI tests.

Before you create an application and continue with this tutorial, it is important to understand the concept of an edit branch and how it is used in DevOps Test.

An **edit branch** is a temporary working branch that provides an isolated workspace for creating or modifying applications, tests, and test suites without impacting the master branch. Using an edit branch allows you to safely make, review, and validate changes before merging them into the master branch.

In this task, you will use an edit branch to create an application that you can use to record and manage Web UI tests.

Now, you can begin with the task after logging in to DevOps Test and selecting the project. For example, *myproj*.

1. Click **Author > Test Editor**.
   The **Test Editor** page and **Navigator** panel are displayed.
2. Create an application by performing one of the following actions:
   - Click the **Action Menu** icon and select **Application**.
   - Right-click the project repository in the Test Navigator panel and select **New Application**.
   - Click **Create Application** on the **Test Editor** page. This option is available only if no applications exist in the project.

   If you are working on the **master** branch, the **Edit Branch** dialog is displayed.
3. Perform the following action to create and edit a branch and use it throughout this tutorial:
   a. Click **Create a new edit branch**.
   b. Enter a branch name. For example, *HRM_branch*
   c. Click **Save**.

   You have created the *HRM_branch* successfully, and the **Create Application** dialog is displayed.
4. Enter a unique name in the **Name** field. For example, *HRM_application*.
5. Click **Create**.

   The application is successfully created and the **Details** tab opens for further configuration.
6. Enter the URL of the HR management application. For example, *https://opensource-demo.orangehrmlive.com/web/index.php/auth/login*.
7. Enter a summary or relevant details about the application in the **Description** field.
8. Click the **Save** icon to save the details.

Results

You have created an application successfully.

You have completed Lesson 1.

What to do next

In the next lesson, you will learn to create a Web UI test. See Lesson 2 on page 711.

Lesson 2: Create a Web UI test by recording user actions

In this lesson, you will learn how to record a Web UI test by capturing screens and user actions on the web pages.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

In this task, you will record user actions on the login page and add employees to the employee list in the HR management application to create a Web UI test by using DevOps Test.

1. In the **Test Editor** page, perform any one of the following actions to record a test:
   - Click the **Action** menu icon in the **Navigator** panel to open the menu and click the **Test** option.
   - Click the **Create Test** icon on the **Details** tab.
   - Click **Create asset > Create Test** on the **Test Editor** page.
   - Right-click the created application in the **Navigator** panel, select **New > Test**.

   The **Create Test** dialog is displayed.

   In the **Details** tab, ensure that the correct application is selected and the location is set.
2. Enter the test asset name in the **Asset Name** field. For example, *HRM_add_employee_test*.
3. Click **Next**.
   The **Source** tab is displayed.
4. Select the **By recording** option in the **Select source** drop-down list to record a test.
5. In the **Agent** section, select the installer package for your operating system.

   The following installer options are available:
   - **Windows (64-bit .msi)**
   - **Mac OS (64-bit .zip)**
   - **Debian/Ubuntu (64-bit .deb)**
   - **RHEL/Fedora/OpenSUSE (64-bit .rpm)**
6. Click **Download**.
7. Click the downloaded agent file to install the agent.
8. Click **Proceed** to accept the download of the proprietary certificate if prompted.

   For detailed installation and upgrade steps depending on the installation status of the agent, refer to the agent installation table on page 773.
9. Click **Verify** to confirm the agent installation.
10. Select a browser from the **Browser** drop-down list.
11. Click **Record** to launch the browser and start recording the test.

    A new browser window is displayed, and the application that you selected for recording is opened.
12. Perform user actions on the Login and Add Employee page of the HRM application. For example, interacting with the user name field, password field, and login button.

    You must allow a delay of a few seconds between user actions.

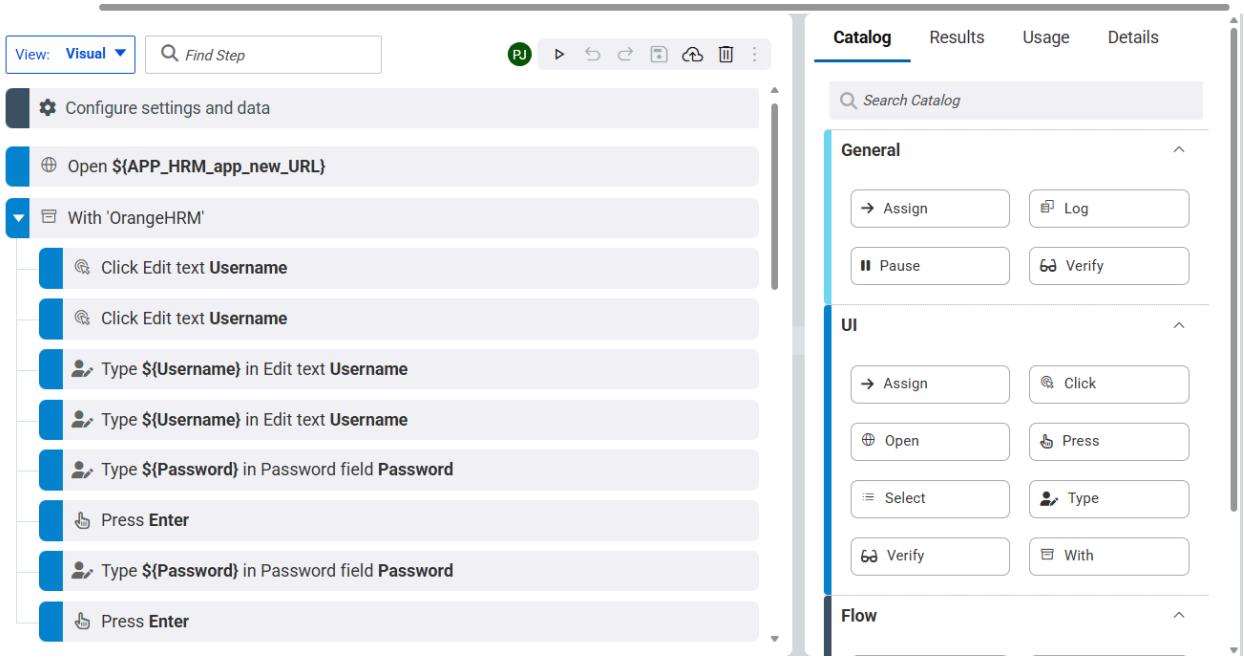
13. Close the browser to stop the recording.

    The following message is displayed in the **Create Asset** dialog.

    `The asset is now ready for your review`.

14. Click **Done**.

    The *HRM_add_employee_test* test asset is created, and the recorded steps are displayed in the **Test Editor.**



The test is displayed in the **Visual** view, which is the default view. You can switch to the **Code** view to view all the step details.

The test captured interactions with the following fields and controls:

- **First Name**
- **Middle Name**
- **Last Name**
- **Employee ID**
- **Create Login Details** toggle
- **Username**
- **Status** (Enabled)
- **Password**

- **Confirm Password**
- **Save** button

Results

You have successfully recorded the user actions for the HRM application and created a test asset.

You have completed Lesson 2.

What to do next

In the next lesson, you will learn how to add a variable to recorded Web UI test steps. See Lesson 3 on page 714.

Lesson 3: Create and add variables to a Web UI test

In this lesson, you will learn how to create and add test variables to store values and reuse them across test steps. By using variables, you can manage test data centrally and apply it dynamically during test execution.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

In the recorded test, the Employee ID value was manually entered as `012345`. In this task, you will create a test variable to store a new Employee ID `23456` and modify the recorded test step to use this variable instead of the manually entered value. This method helps make the test more flexible and reusable by separating test data from test logic.

1. In the test steps list, click **Configure settings and data**.
2. In the **Variables** section, click **Create Variable**.
3. Enter a variable name. For example, *new_EmployeeID*.
4. Enter the required value in the **Value** field. For example, *23456*.
5. Save the variable by clicking outside the **Value** field.

   The newly created variable is listed in the **Variables** section.
6. Click the **Close** icon to close **Configure settings and data**.
7. Perform the following actions to replace the Employee ID value in the recorded steps:

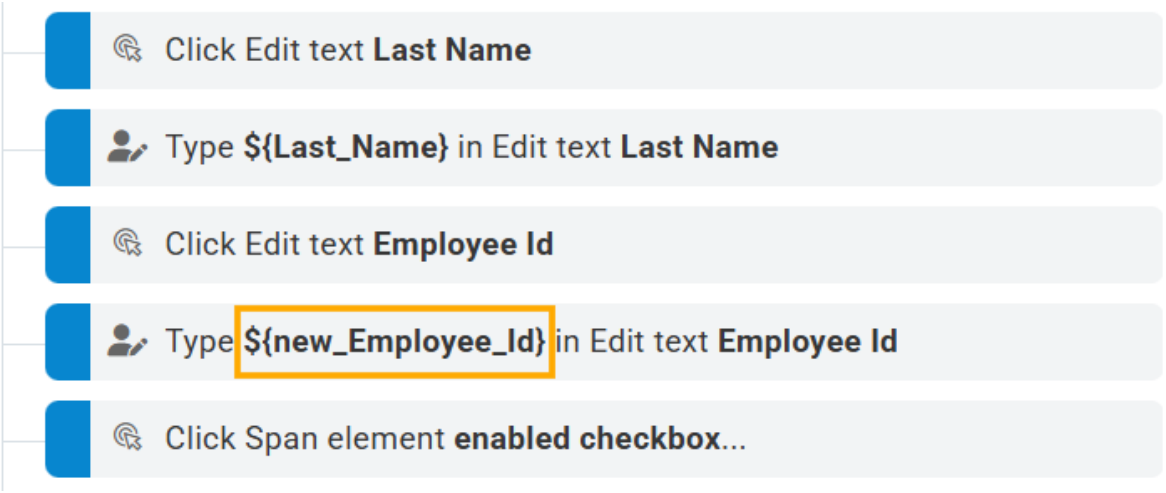
   a. Locate the recorded step that enters the Employee ID value (currently *012345*).

      The step opens in the **Details** tab.

   b. In the **Value** field, replace the manually entered Employee ID with the variable. For example, *${new_EmployeeID}*.

   c. Close the **Details** tab.

The **Employee ID** step is modified to use a ${new_EmployeeID} value, as shown in the image.



8. Click **Save** to save the modified step.

Results

You have now added the variable to the recorded step.

You have completed Lesson 3.

What to do next

In the next lesson, you will learn how to modify the test steps. See Lesson 4 on page 715.

Lesson 4: Modify test steps

In this lesson, you will learn how to modify an existing Web UI test from the **Test Editor** view.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

In this task, you will learn how to modify an existing recorded test step by changing the action from *Click* to *Double click* by using the **Smartshot** view, without re-recording the test.

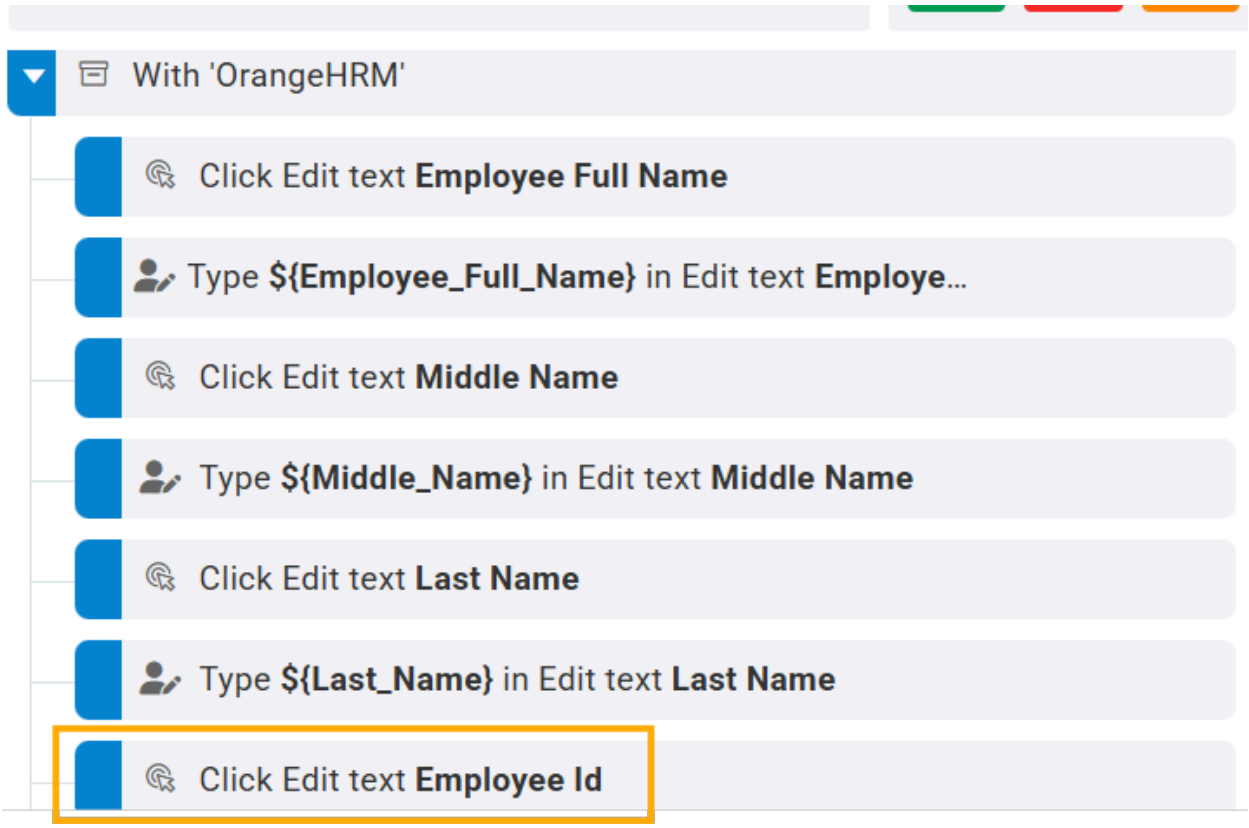
User interface changes in an application can alter the behavior of recorded test steps. For example, after a UI update, the *Employee ID* field might require a *double-click* instead of a single click to enable editing. As a result, a previously recorded *Click* action might no longer work as expected.

You will also learn how to update the target UI element and refine object identification to ensure the test continues to work correctly after UI changes.

Smartshot editing is useful when the **target element** is modified due to UI updates. A **target element** is a UI element in the application that a test step interacts with during a test run. For example, in the step *Click Edit text Employee Id*, the *Employee Id* field is the target element.

1. In the recorded test steps list, locate the *Click Edit text **Employee Id*** step and click it. For example, *Edit text – Employee Id*.

   The **Smartshot** view opens for the selected step.



2. Click the **Expand view** icon ← to expand the Smartshot view for better visibility.
3. Click the *Employee Id* field in the Smartshot. The field is highlighted now in the Smartshot.
4. Right-click the *Employee Id* field now and select **Use this element as step** target:

   a. Use this step when the Employee ID field is not captured correctly in Smartshot.

   b. This updates the step target with the correct UI element and properties before you modify the action.

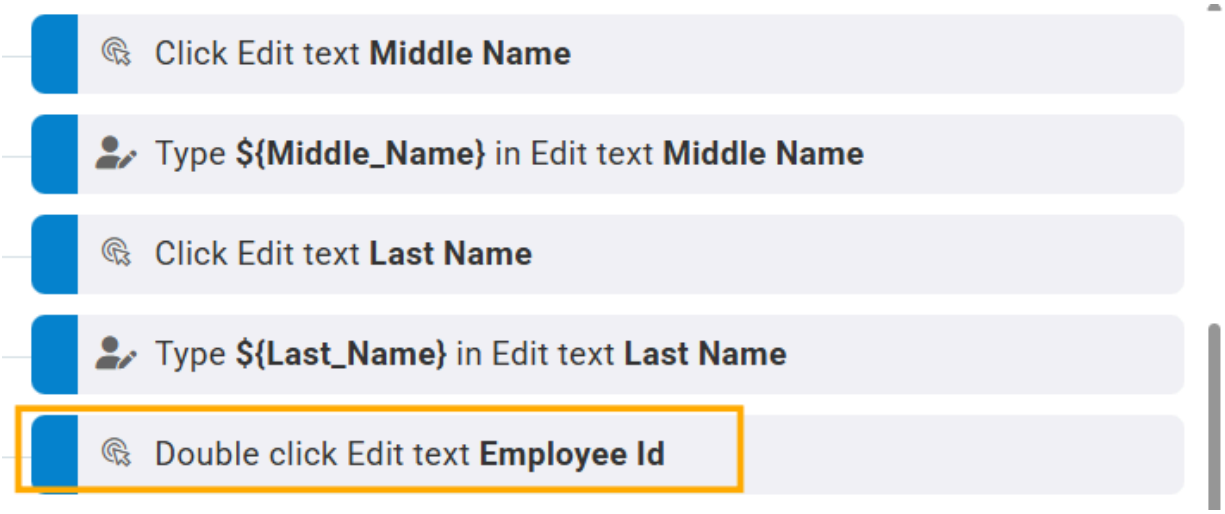
5. In the **Details** tab, review the following current configuration:

   - **UI Object**: The type of UI element. For example, **Edit text**.
   - **Action**: The action performed on the element. For example, **Click**.

6. Select a **Double Click** from the **Action** list as required by your test scenario to modify the **Action**.
7. In the **Elements** panel, you can browse and select the correct UI element.

8. Verify the selected element details in the **Properties** panel to ensure attributes like enabled, exist, and tagName match the expected UI state.
9. Click the **Close** icon to close the **Smartshot** view.

   The *Employee ID* step is modified to use a *Double click* action, as shown in the image.



10. Click **Save**.

Results

You have modified the behavior of the recorded tests using Smartshots.

You have completed Lesson 4.

What to do next

In the next lesson, you will learn how to manage steps using the right-click menu. See Lesson 5 on page 717.

Lesson 5: Manage test steps

In this lesson, you will learn how to modify and manage recorded Web UI test steps by using the right-click menu to reorder, reuse, disable, or delete steps.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

This task demonstrates how to manage recorded Web UI test steps by using the right-click menu. You will learn how to reorganize test steps, reuse existing steps, temporarily disable steps, and remove unnecessary or duplicate steps to keep your test flow clean and maintainable.

The examples in this task use the *Employee ID* step to show how these actions can be performed while editing an existing Web UI test.

1. Right-click on the *Click Edit text – Employee Id* step and click **Select**.

   Checkboxes appear next to the steps.
2. Select multiple steps, such as duplicate *Click Edit text – Employee Id* steps.
3. Right-click the selected steps to open the context menu.

   Once the steps are selected, additional options such as **Extract Steps**, **Deselect All**, and **Disable** become available.
4. Right-click the duplicate *Click Edit text – Employee Id* step and click **Disable** to disable it to temporarily skip it during test run.

   Alternatively, click the *Click Edit text – Employee Id* step, click the **Advanced** tab, and select the **Disable** checkbox.
5. Right-click the disabled step and click **Enable** to enable the step again.
6. Use the **Move Up** and **Move Down** options to ensure that the *Click Edit text – Employee Id* occurs at the correct point in the test sequence. You can reorder the steps using the following options:
   - Right-click a selected step and click to move the step earlier in the test flow.
   - Right-click a selected step and click to move the step later in the test flow.
7. Right-click the *Click Edit text – Employee Id* step and click **Copy**.
8. Right-click another step and click **Paste**.

   The copied Employee Id step is pasted immediately after the selected step.
9. Right-click the extra step and click **Delete** to permanently remove it from the test.
10. Click **Save**.

Results

Duplicate steps are disabled or removed, steps are reordered and copied as needed, and the test is updated with the new step sequence.

You have completed Lesson 5.

What to do next

In the next lesson, you will learn how to add a new test step from the **Catalog**. See Lesson 6 on page 718.

Lesson 6: Add steps from the Catalog to a Web UI test

In this lesson, you will learn how to add test steps from the catalog to the recorded Web UI test to quickly select and reuse steps based on your testing needs.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

In this task, you improve the reliability of your Web UI test by validating critical input data before allowing the test to proceed. The catalog groups commonly used actions into **General**, **UI**, and **Flow** categories, helping you quickly select the right action based on your testing requirement.

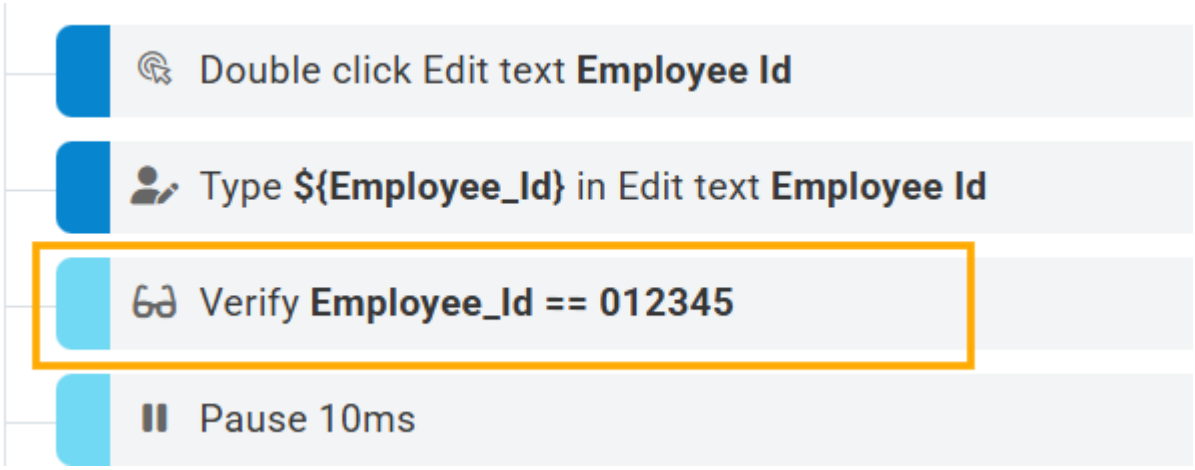
First, you verify that the *Employee_Id* matches the expected value by adding a **Verify** step from the **General** catalog. This verification ensures that the test is using the correct employee data. If the *Employee_Id* does not match, the verification step fails, and the failure is recorded in the test results.

Next, you add an **If** condition from the **Flow** catalog to validate the password value on the Add employee page. The steps that confirm the password and save the employee data will run only if the password meets the expected condition. If the condition is not met, the steps inside the **If** block are not skipped, preventing the test from continuing with invalid data.

The following catalog categories are used in this task:

1. **General Catalog** - Provides steps used for general operations such as assigning variables, logging information, pausing execution, or verifying values.
2. **UI Catalog** - Provides steps to interact with user interface elements, like clicking buttons, typing text, selecting dropdowns, or opening web pages.
3. **Flow Catalog** - Provides steps for controlling the flow of the test, such as loops, conditional logic, or handling errors.

1. Locate the recorded step that enters the *Employee_Id* value.
2. From the **General** catalog, right-click **Verify** and select **Add step**.
   Alternatively, drag and drop the **Verify** step from the **General catalog** and place it below the selected test step.
3. Click the **Verify** step, and configure the following **Verification points**:
   a. Select the variable *Employee_Id*.
   b. Set the comparison operator to **Equals**.
   c. Select the expected variable. For example, *Employee_ID*, or enter the expected value (*012345*, the value that was entered while recording the test).



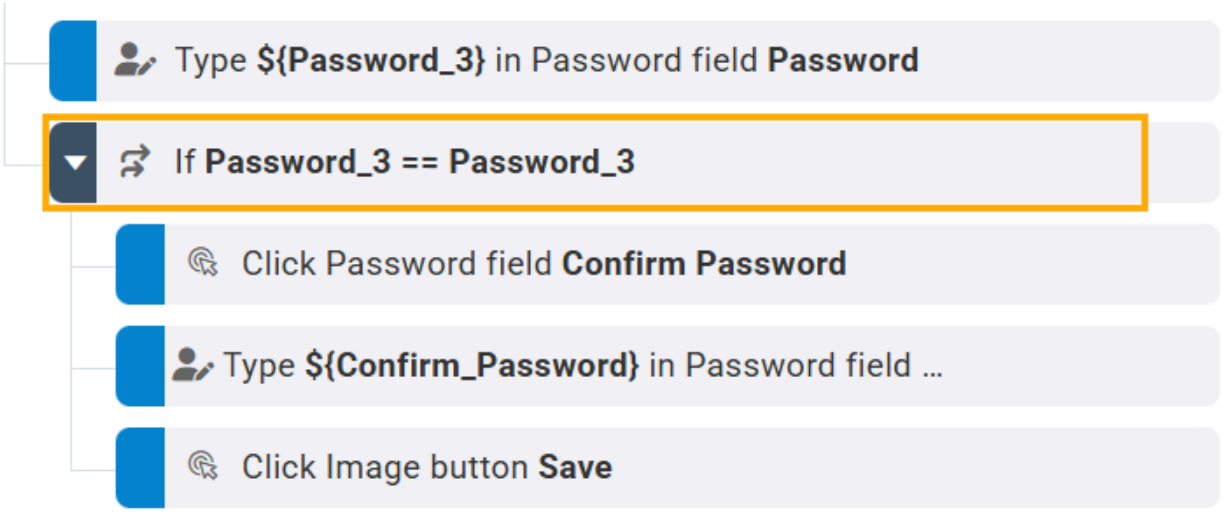
Note: If you select a different variable or enter a different value, this verification step fails while running the test.

4. From the **Flow** catalog, add an **If** step after the *Type Password in Password field* step.
5. Drag the remaining steps, such as *Confirm Password* and *Save*, into the **If** block.
6. Click the **If** step and configure the following details in the **Condition** section:

    a. From the first drop-down list, select the password variable *Password_3*.

    b. From the operator drop-down list, select **Equals**.

    c. In the value field, select the expected password variable, or enter the expected password value.
    If you select a different variable or enter a value that does not match the expected password, the **If** condition fails. When the condition fails, the steps inside the **If** block will not run, and the remaining steps stop running.



7. Click **Save** to save the test.

The following table lists all catalog elements available in DevOps Test, including **General**, **UI**, and **Flow** catalogs. Use these catalogs to try adding new test steps by selecting the actions that best fit your scenario.

Catalog	Action	Description
General	Assign	Assigns a value to a variable
	Verify	Verifies values, conditions, or expressions
	Log	Logs messages during the test run
	Wait	Pauses the test for a specified duration
UI	Click	Performs a click action on a UI element
	Double Click	Performs a double-click action on a UI element

Catalog	Action	Description
	Type Text	Enters text into an input field
	Select	Selects a value from a list or dropdown
	Verify	Verifies UI properties such as text, value, or state
Flow	If	Runs steps conditionally based on an expression
	Else	Runs steps when the If condition is false
	End If	Marks the end of an If block
	Loop	Repeats steps based on a condition
	Break	Exits a loop

Results

You have successfully added new test steps by using the catalog.

You have completed Lesson 6.

What to do next

In the next lesson, you will learn how to add Verification points using Smartshots. See Lesson 7 on page 721.

Lesson 7: Add verification points to test steps by using Smartshots

In this lesson, you will learn how to add verification points to the test steps by using the **Smartshots** to validate UI elements, values, and messages during a test run.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

In this task, you add a verification point to check whether the *Create Login Details* field is *enabled*. Verification points are evaluated when the test runs.

As each test step is performed, the following associated validations are verified:

- If the validation condition is met, the test proceeds to the next step.
- If the validation condition is not met, the step is marked as failed, and the failure is recorded in the test results.

Quick rules to remember

Use the following rules to choose the correct **Verify** action and verification point:

Table 588. Validation options

Validation requirement	Recommended Verify action
Field value displayed on the screen	Use the Verify action from the UI Catalog to validate values displayed in the UI.
Mandatory field or error message	Use the Verify action from the UI Catalog to check UI-level validation messages.
Length or format validation (for example, 5-digit Employee ID)	Use the Verify action from the General catalog to validate length and format rules.
Logical or data-based rules	Use the Verify action from the General catalog to evaluate conditions or business logic.

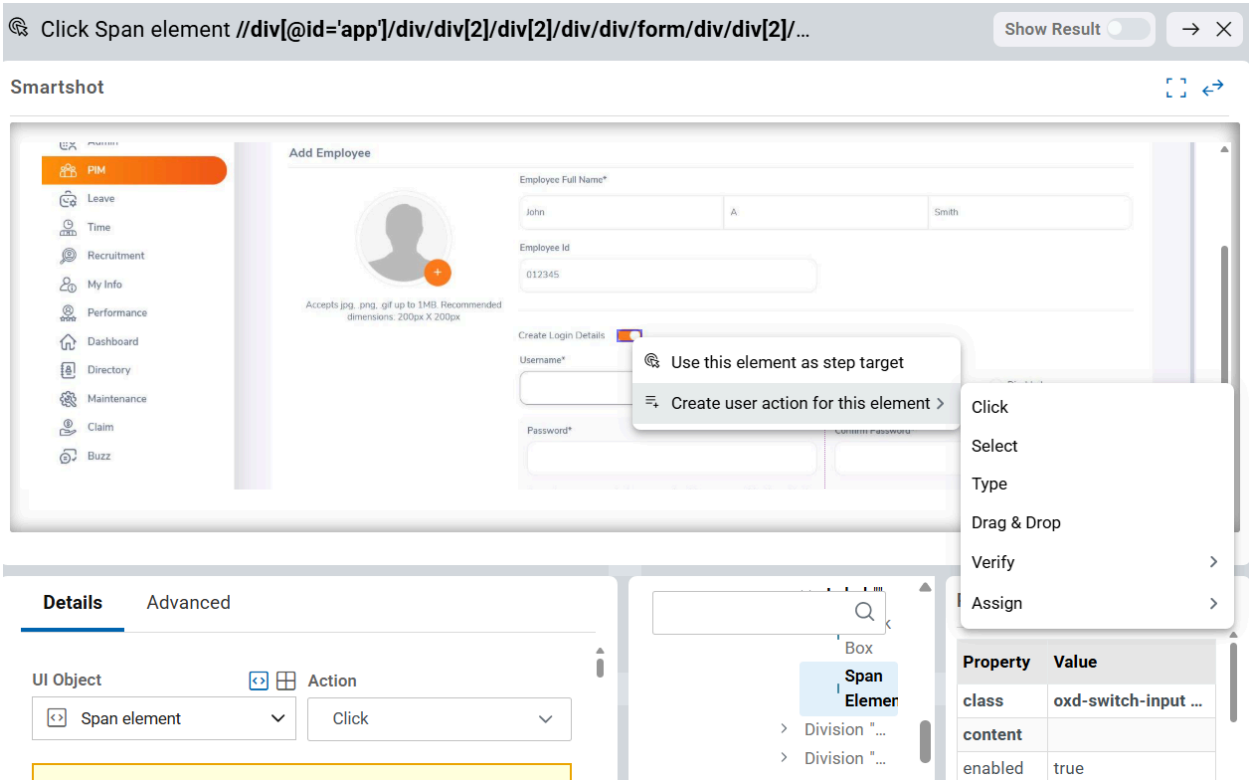
Table 589. Common UI elements and verification points

Element type	Verification point
Label text	`content`
Input field value	`value`
Field presence	`exist`
Enabled or disabled state	`enabled`

1. Locate the recorded step *Create Login Details*, and then click it.

   The step opens in the **Smartshot** view tab.

2. Click the **Expand view** icon ← to expand the Smartshot view for better visibility.
3. Select the *Create Login Details* toggle field.



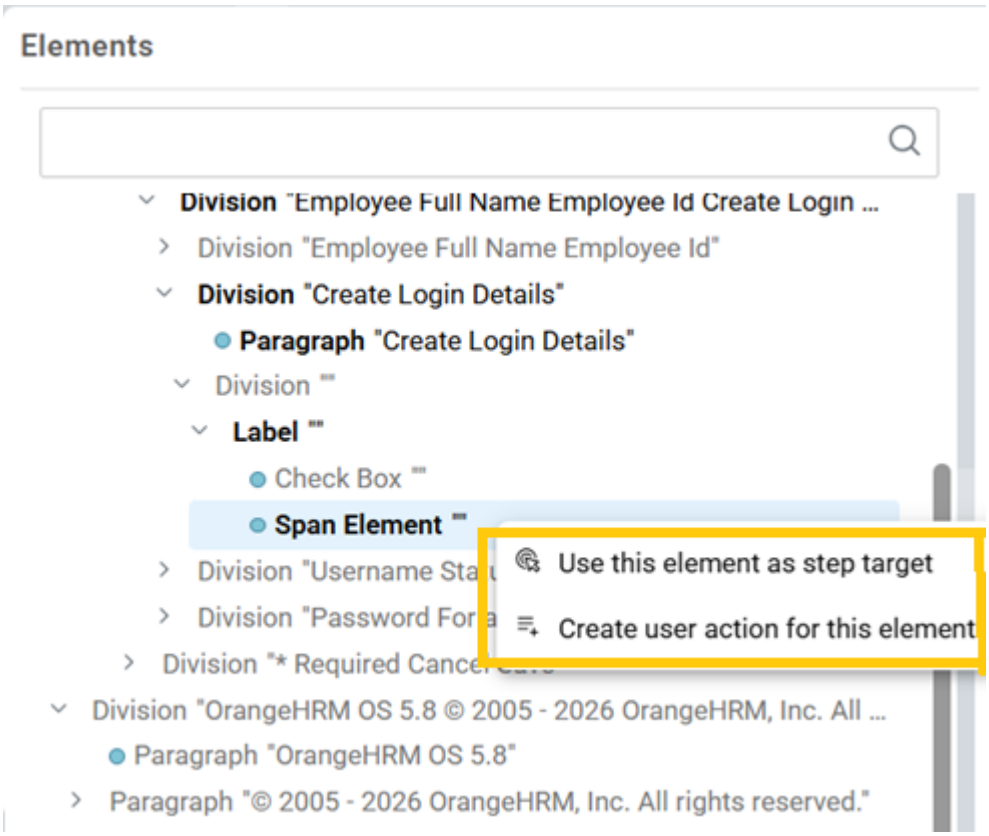
4. Right-click and select **Create user action for this element > Verify**. This allows you to add a verification that checks a specific property of the selected UI element. In this scenario, this step creates a verification point to check whether the **Create Login Details UI** element is **enabled**.
5. Select **enabled:true** option from the displayed **Verification points** list.
6. You can veiw the created the **Verification** in the **Details** tab.
7. Notice that once the UI element is selected in **Smartshot**, the same element is automatically highlighted in both the **Elements** panel and the **Properties** panel.

    - The **Elements** panel shows the UI elements captured in the recorded test.
    - The **Properties** panel displays the properties of the currently selected UI element.

    You can use either panel to add the same **Verify** step.

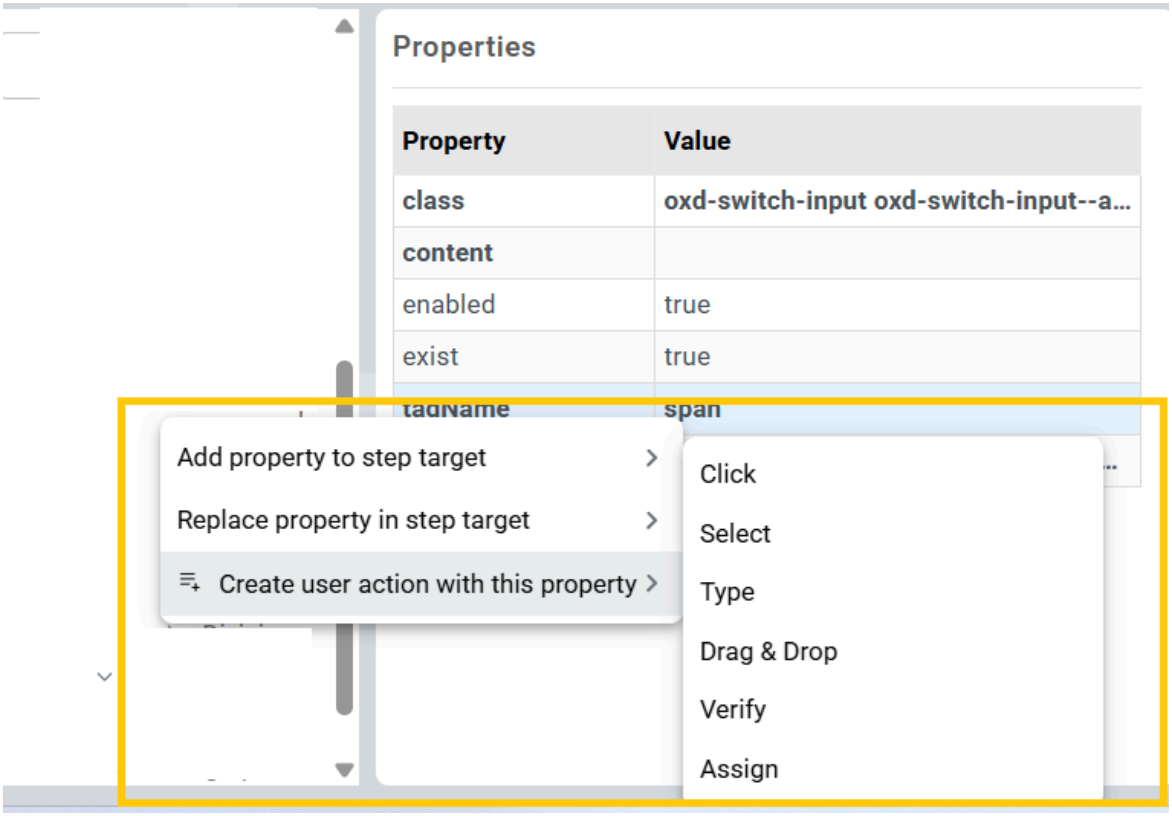
    - From the **Elements** panel, right-click the highlighted UI element and select **Create user action for this element > Verify**. For example, *Span Element* in the **Elements** panel.

      You are creating a verification based on a specific property of the UI element, rather than the entire element.

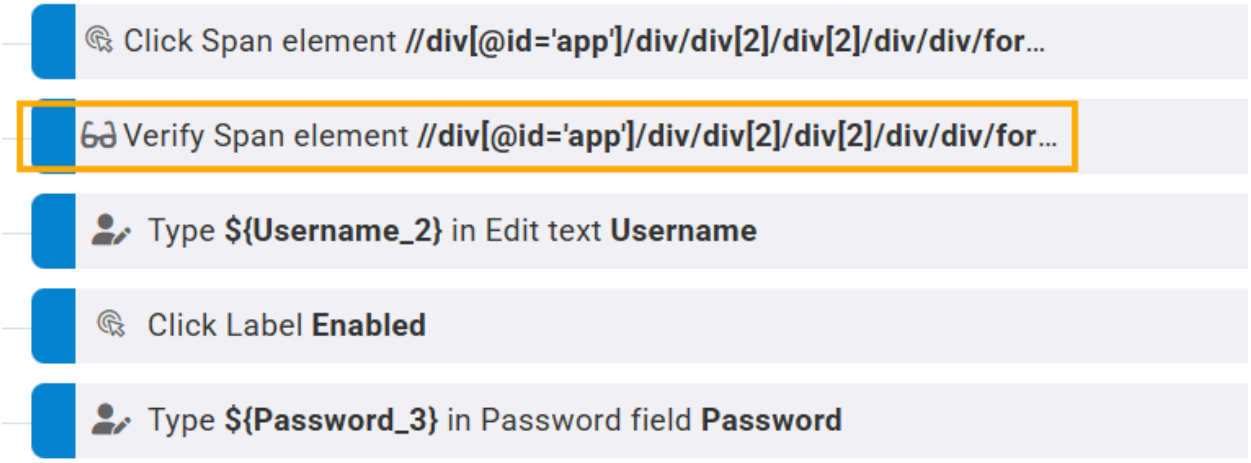


◦ From the **Properties** panel, right-click the **tagName** property and select **Create user action with this property > Verify**. For example, **span**.

You are creating a verification based on a specific property of the UI element, rather than the entire element.



8. Click **Close** to close the **Smartshot**.
   The **Verify** step is added to the recorded steps.



9. Click **Save** to save the modified test.

Results

You have added verification points to the recorded *Create Login Details* step.

You have completed Lesson 7.

What to do next

In the next lesson, you will learn how to run the test. See Lesson 8 on page 726.

Lesson 8: Run the Web UI test

In this lesson, you will learn how to run a recorded Web UI test in DevOps Test and verify that the web application performs as expected.

Before you begin

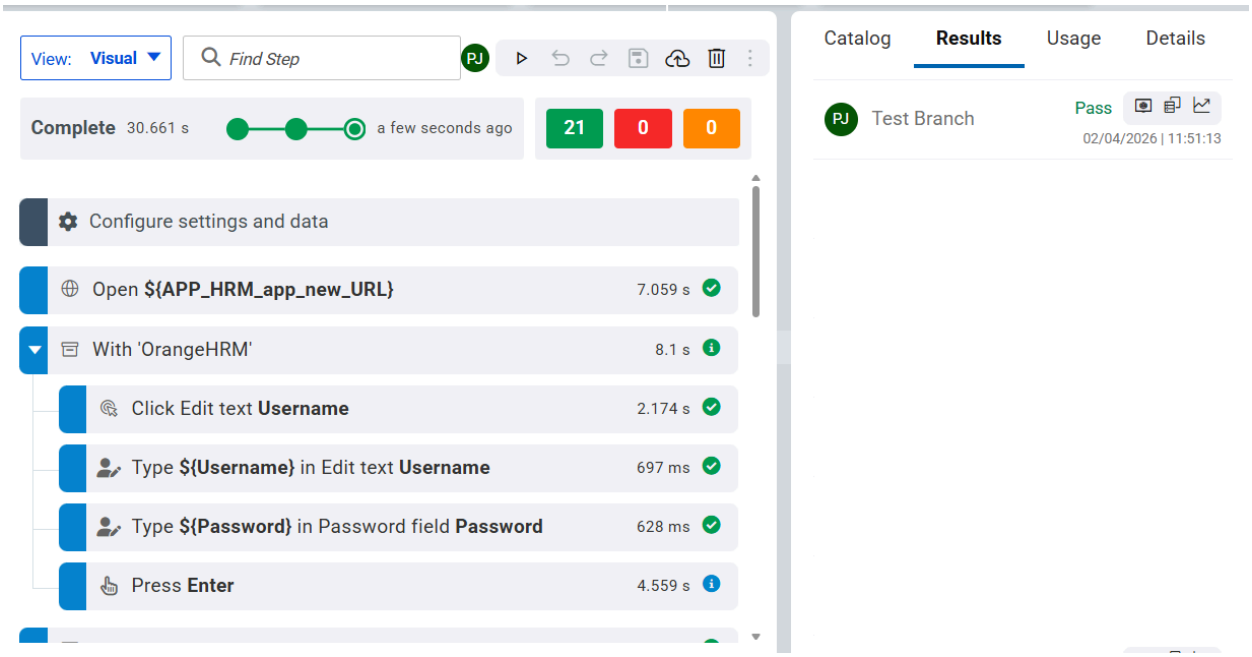
Ensure that you have completed the previous lessons in this tutorial.

About this task

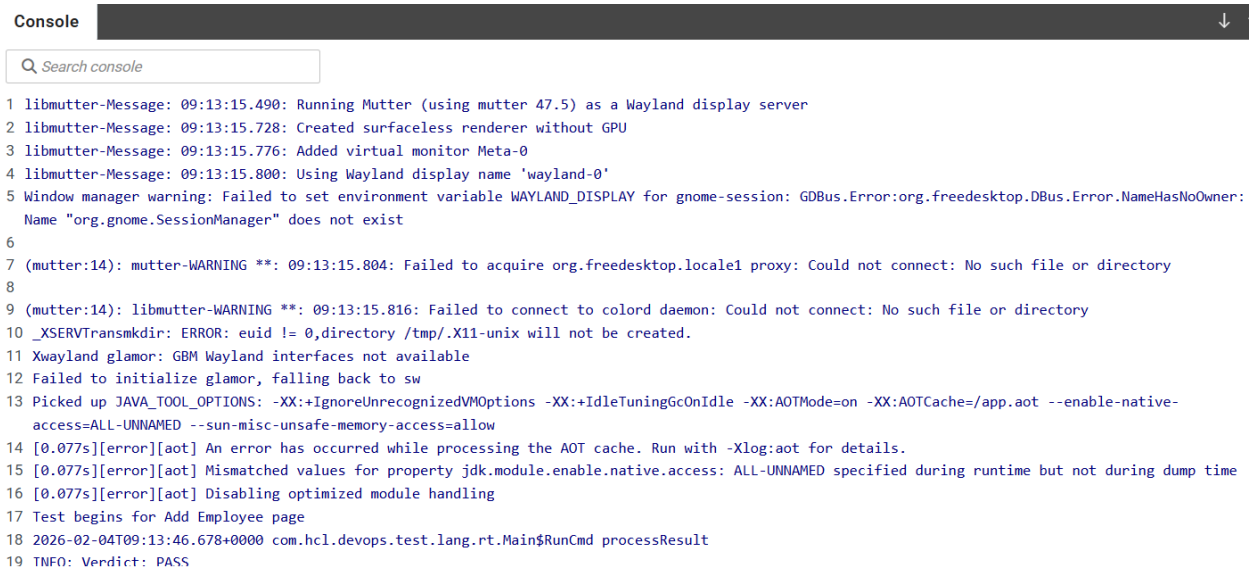
In this task, you run the recorded Web UI test to simulate creating a new employee in the HR management application. This test run validates that key workflows such as Employee Id generation, role assignment, and data saving work correctly.

1. Click the icon, and then select the **Run on Default Cluster** option to start the test run for validating the recorded test for the HR management application.

   When the test run starts, the **Catalog** tab automatically switches to the **Results** tab so that you can monitor the execution status.
2. Observe the following stages during the test run on the **Test Editor** page:
   - **In transition**: The test setup is in progress.
   - **Running**: The test run is active.
   - **Complete**: The test run is completed. This section also displays the time taken for the test run.
   - The passed count, failed count, and error count are displayed on the **Test Editor** page.



3. Click the **Console** tab to view information such as error messages, logs, and the test verdict.
   You can then use this output to troubleshoot issues by reviewing details, such as session IDs and error messages.



   At any point, you can stop the test manually by clicking the **Stop playback** icon ☐. When stopped, the test status is displayed as **Stopped by User** on the progress bar and **Inconclusive** on the **Results** tab.

Results

You have successfully run the recorded Web UI test and viewed the test execution results.

You have completed Lesson 8.

What to do next

In the next lesson, you will learn how to view and analyze the test results. See Lesson 9 on page 728.

Lesson 9: View and analyze test results

In this lesson, you will learn how to view and analyze a **Functional Report** after running a Web UI test on the HR management application. You will also learn how to investigate a test failure using a real failure scenario.

Before you begin

Ensure that you have completed the previous lessons in this tutorial.

About this task

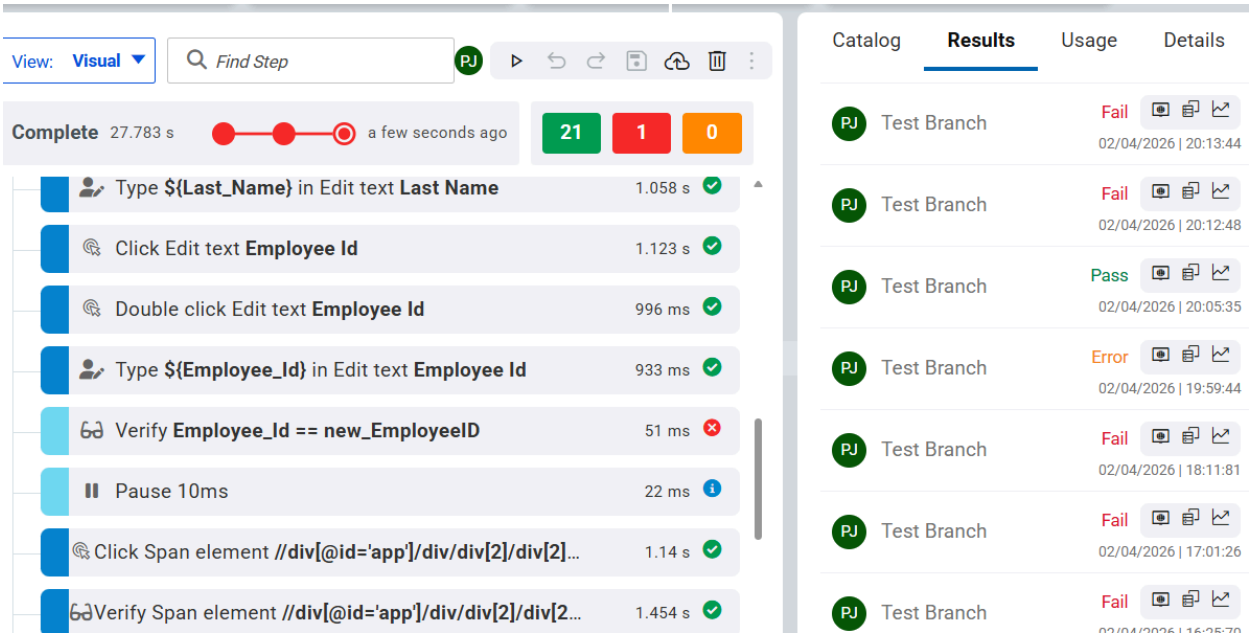
In this task, you will review a failed test run to understand why the failure occurred and identify the failed test steps. You analyze the **Functional Report** and **Test Log** generated by DevOps Test after the test run.

The **Functional Report** provides a comprehensive view of the test, including overall test status, test environment details, performed test steps, validation results, and screenshots captured during the test run. The **Test Log** shows a detailed, step-by-step record of what happened while running the test.

The Functional Report and Test Log help you confirm whether HRM application workflows, such as employee creation and verification, worked as expected and quickly pinpoint the cause of failures. The example used in this task demonstrates a test failure caused by an *Employee_Id* value mismatch.

1. Click the icon, and then select the **Run on Default Cluster** option to start the test for validating the recorded test for the HR management application.

   In the result summary, observe that the test is marked as 1 failed count.



2. Click the **Functional Report** icon to open the report in a separate window.

3. Click the **Expand** icon in the **Functional Report** to view detailed results.

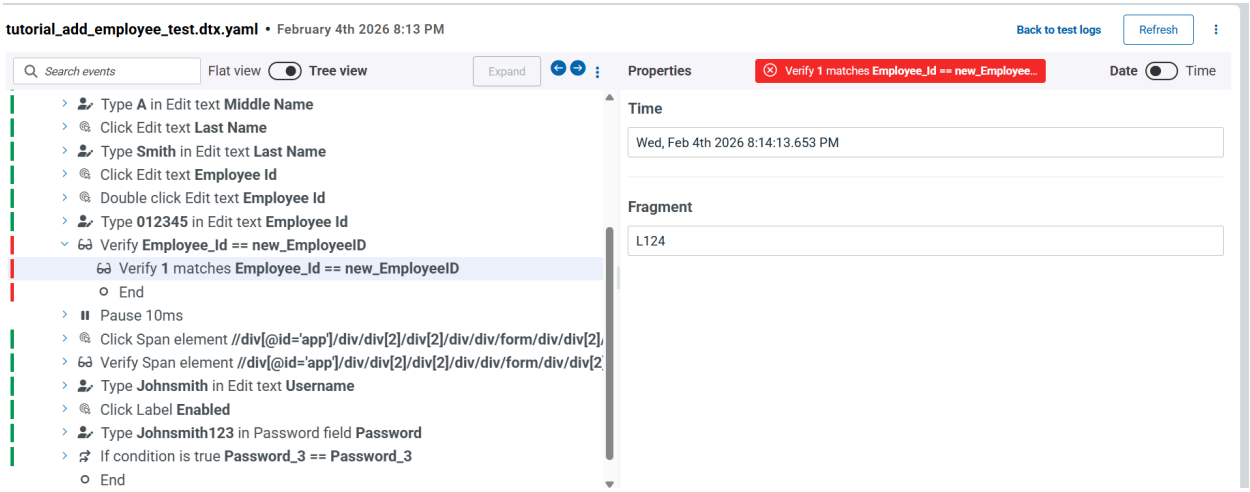
   You can now see the failed step and identify the reason for failure.

4. Click the **Open in new tab** icon to access the specific test in the **Test Editor** page, where you can review and edit the test steps.

   Review the failed step.

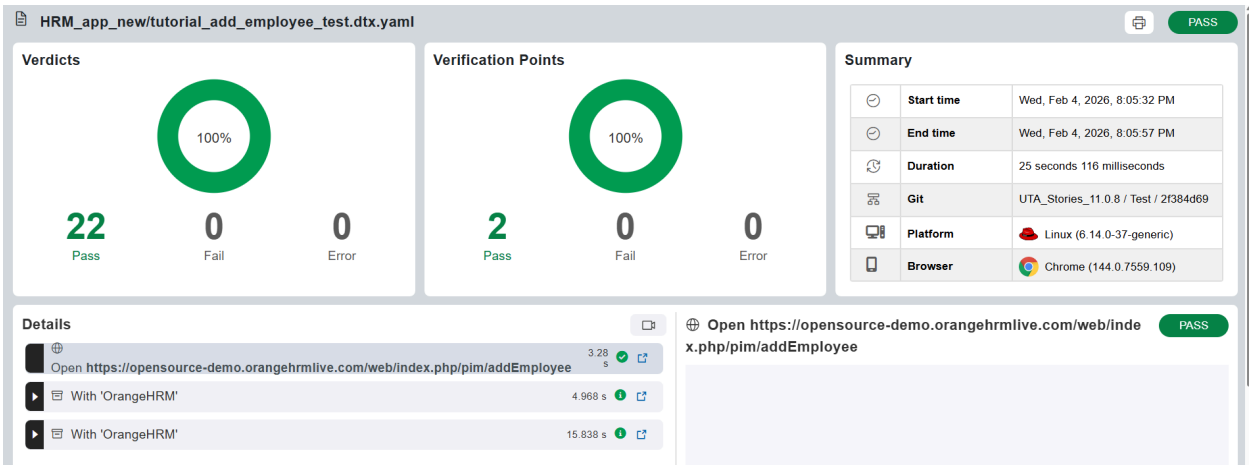
   The *Employee_ID* step failed because the value of *Employee_Id* is defined as *new_Employee_Id* in the **Verify** step.

5. In the **Verification Points**, select the correct variable value (*Employee_Id*) from the drop-down list.
6. Click the **Test Log** icon to open the **Test Log Report** to view the step-by-step details of each test step while running the test.



7. Click **Save** to update the test.
8. Click icon, and then select the **Run on Default Cluster** option to run the test again.

   - The test completes with **0 failed count**
   - The test status shows **Pass** in the Results tab
   - The Functional Report shows **100% Pass**



9. Click the **Publish** icon in the **Test Editor** page to publish the changes from the *HRM_branch* edit branch to a *master* branch in the project repository.

   After publishing, the changes are saved, the edit branch is automatically deleted, and is removed from the **Branch** drop-down list and the remote Git repository.

Results

You have successfully viewed and analyzed the Functional Report and Test Log, fixed the failure, and rerun the test with a **Pass** status.

You have successfully completed this tutorial.

Administrator Guide

This guide describes how to install the Test software.

After you install the software, you can perform administration tasks such as license configuration, user management, security, memory and disk usage management, back up and restore user data, and other tasks that a server administrator can perform. This guide is intended for administrators.

As a *Server Administrator*, you can assign the role of a *Server Administrator* to any user. For information about how to change the role, see Default user administration.

Remember: When your role is changed from a user to a *Server Administrator*, you must log out and log in again so that Test can apply the change in your role.

Configuration of DevOps Test

You can find information about the tasks that you must perform after you installed Test to configure Test.

The following table describes the tasks that you can perform as an administrator after you installed Test:

Require-ment	Tasks	More information
Required	Review the default user management to know how Test manages the users and their authentication.	User administration
Required	Import Certificate Authority (CA) into a browser to prevent certificate errors when you access the Test UI.	Certificate authority: Importing and extending lists on page 732
Optional	Copy the third-party application Jars to Kubernetes to run API Suites that uses a transport.	Copying third-party application Jars to Kubernetes on page 736
Optional	Configure multi-factor authentication (MFA) for log-ging into Test.	
Optional	Change the password seed to provide enhanced se-curity to Test.	Changing the password seed
Optional	Disable the server extensions that are enabled during the installation of Test, if you do not want the serv-er extensions to run on the cluster to save server re-sources.	Disabling server extensions on page 739

Management of third-party licenses

When you want to use built-in web browsers such as Google Chrome or Microsoft Edge, you must first accept the third-party license agreement of these browsers. You can manage third-party license agreements on the **General Settings** page in Test .

You can find the following information:

- Accepting third-party licenses on page 731
- Removing acceptance of third-party licenses on page 732

Accepting third-party licenses

You can accept any third-party license agreements on the **General Settings** page in Test .

About this task

Test prompts you to accept the third-party license agreements when you sign in or log in.. If you do not accept the third-party license agreements, Test continues to prompt whenever you log in.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Account** icon, and then click **Account Settings**.
   Result
   The **General Settings** page of your account is displayed.
3. Click the **Terms of Service** links in the **Third-Party Agreements** section to review the license agreements and terms of use for Google Chrome and Microsoft Edge.
4. Perform any of the following actions:
   - Select the checkbox for the browser that you want to use to accept the license agreement.
   - Select both **Google Chrome** and **Microsoft Edge** checkboxes, if you want to use both the browsers.
5. Click **Save**.

Results

You accepted the license agreements for either Google Chrome or Microsoft Edge browser or both the browsers.

Removing acceptance of third-party licenses

You can remove the acceptance of third-party licenses when it is no longer required.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **User** icon, and then click **Account Settings**.
   Result
   The **General Settings** page of your account is displayed.

   The **Third-Party Agreements** section displays checkboxes for **Google Chrome** and **Microsoft Edge** options.
3. Perform any of the following actions:
   - Clear the checkbox for the browser that you do not want to use.
   - Clear both the **Google Chrome** and **Microsoft Edge** checkboxes, if you do not want to use both the browsers.
4. Click **Save**.

Results

You removed your acceptance of the license agreements for either Google Chrome or Microsoft Edge browser or both the browsers.

Certificate authority: Importing and extending lists

When you want browsers and other applications to access Test , you must enable browsers to trust the certificate authority (CA) of Test. For some applications that integrate with Test, you must also extend the CA lists that are trusted by the applications.

Before you import CA and extend the CA list, you must save the CA assigned to Test. See Saving the certificate authority on page 733.

You can find the following tasks to import the certificate authority in the browser that you want to use:

- Importing the certificate authority into the Google Chrome browser on page 734
- Importing the certificate authority into the Microsoft Edge browser on page 734
- Importing the certificate authority into the Mozilla Firefox browser on page 735

You can find the following tasks when you want to extend the CA lists:

- Extending the trusted certificate authority list on Linux on page 735
- Extending the trusted certificate authority list on Windows on page 736

Saving the certificate authority

You can save the certificate authority (CA) when you want to import the CA or extend the CA lists that are trusted by the applications.

Before you begin

You must have completed the following tasks:

- Installed Test on a Kubernetes or OpenShift environment.
- Ensured that you have permissions to view the secrets that are created during installation of Test on a Kubernetes or OpenShift environment.

1. Access the Kubernetes or OpenShift cluster where you installed Test.
2. Run the following command to save the CA in `ca-master.crt` file.

```
kubectl get secret -n devops-system ingress -o "jsonpath={.data['ca\.crt']}" | base64 --decode >
 ca-master.crt
```

Remember: `devops-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the `devops-system`.

Results

You have successfully saved the CA.

What to do next

You can perform the following tasks:

- Import the CA into the Google Chrome browser. See Importing the certificate authority into the Google Chrome browser on page 734.
- Import the CA into the Microsoft Edge browser. See Importing the certificate authority into the Microsoft Edge browser on page 734.

- Import the CA into the Mozilla Firefox browser. See Importing the certificate authority into the Mozilla Firefox browser on page 735.
- Extend the trusted CA list on Windows. See Extending the trusted certificate authority list on Windows on page 736.
- Extend the trusted CA list on Linux. See Extending the trusted certificate authority list on Linux on page 735.

Importing the certificate authority into the Google Chrome browser

You must import the certificate authority (CA) into the Google Chrome browser to prevent certificate errors when accessing the Test .

Before you begin

You must have completed the following tasks:

- You must have saved the CA assigned to Test. See Saving the certificate authority on page 733.
- Installed the browser that you want to use to access Test.

1. Open the Google Chrome browser.
2. Navigate to **Customize and control Google Chrome > Settings**.
3. Click **Security** from the **Privacy and security** pane.
4. Click **Manage certificates**, and then **Import**.
5. Click **Next** in the **Certificate Import Wizard** window.
6. Click **Browse** and select the CA that you want to import, and then click **Next**.
7. Select the **Place all certificates in the following store** option to store the CA securely.
8. Click **Browse**, and then select **Trust Root Certification Authorities** as certificate store.
9. Click **OK**, and then **Finish**.

Results

You have imported the CA into the Google Chrome browser.

Importing the certificate authority into the Microsoft Edge browser

You must import the certificate authority (CA) into the Microsoft Edge browser to prevent certificate errors when accessing the Test .

Before you begin

You must have completed the following tasks:

- You must have saved the CA assigned to Test. See Saving the certificate authority on page 733.
- Installed the browser that you want to use to access Test.

1. Type `certmgr.msc` in the **Start** menu **Search** filed, and then press **Enter**.
2. Expand **Trusted Root Certification Authorities**, and then select **Certificate**.
3. Right-click on the empty space, and then select **All tasks > Import**.
4. Click **Next** in the **Certificate Import Wizard** window.

5. Click **Browse** and select the CA that you want to import, and then click **Next**.
6. Select the **Place all certificates in the following store** option to store the CA securely.
7. Click **Browse**, and then select **Trust Root Certification Authorities** as certificate store.
8. Click **Next**, and then **Finish** to import the certificate.

Results

You have imported the CA into the Microsoft Edge browser.

Importing the certificate authority into the Mozilla Firefox browser

You must import the certificate authority (CA) into the Mozilla Firefox browser to prevent certificate errors when accessing the Test .

Before you begin

You must have completed the following tasks:

- You must have saved the CA assigned to Test. See Saving the certificate authority on page 733.
- Installed the browser that you want to use to access Test.

1. Open the Mozilla Firefox browser.
2. Enter `"about:preferences#privacy"` in the address bar, and then press **Enter**.
3. Locate and click the **View Certificates** option.
4. Select **Authorities** tab, and then click **Import**.
5. Select the CA that you want to import, and then click **Open**.
6. Select **Trust this CA to identify websites** in the **Downloading Certificate** window.
7. Click **OK** to import the CA into the browser, and then close the window.
8. Restart Firefox.

Results

You have imported the CA into the Mozilla Firefox browser.

Extending the trusted certificate authority list on Linux

Test trusts the certificates signed by a recognized certificate authority (CA). When you have any additional CAs, you must extend the trusted certificates list by using an environment variable on Linux.

Before you begin

You must have saved the CA assigned to Test. See Saving the certificate authority on page 733.

1. Open the terminal.
2. Run the following command to create an environment variable on the Linux operating system:

```
export NODE_EXTRA_CA_CERTS= {path of the downloaded CA with the file extension}
```

For example,

```
export NODE_EXTRA_CA_CERTS=/opt/mycert/ca-master.crt
```

Results

You have extended the trusted CA certificates list on Linux.

Extending the trusted certificate authority list on Windows

Test trusts the certificates signed by a recognized certificate authority (CA). When you have any additional CAs, you must extend the trusted certificates list by using an environment variable on Windows.

Before you begin

You must have saved the CA assigned to Test. See Saving the certificate authority on page 733.

1. Right-click the Windows icon from the Windows taskbar, and then click **System**.
2. Click **Advanced system settings** from the **Related setting** section.
3. Click the **Advanced** tab, and then click **Environment Variables**.
4. Click **New** from the **System variables** section.
5. Enter `NODE_EXTRA_CA_CERTS` in the **Variable name** field.
6. Enter the path of the downloaded CA with the file extension in the **Variable value** field.

   For example, if you stored the `ca-master.crt` CA in the `C:\Users\Desktop\cert` location, then enter the **Variable value** field as `C:\Users\Desktop\cert\ca-master.crt`
7. Click **OK**.

Results

You have extended the trusted CA certificates list on Windows.

Copying third-party application Jars to Kubernetes

You can run API Suites in a project on Test . If the API Suite uses a transport and the transport requires third-party application `Jar` files for a successful run, you must ensure that the third-party application `Jar` files are available at the test run time. To achieve this, you must copy the third-party application `Jar` files to the computer where Test is installed on Kubernetes.

Before you begin

You must have server administrator or cluster administrator privileges.

You must have completed the following tasks:

- Identified the third-party application `Jar` files that are required and copied the files. See Test run considerations for API Suites or API tests on page 1059.
- Copied the third-party application `Jar` files from HCL DevOps Test Integrations and APIs (Test Integrations and APIs) to the directory or folder in Kubernetes from where you can run the kubectl commands.

About this task

You can copy the third-party application `Jar` files to the folder that is specific for the application under the `/data/<application_name>` folder. You need not extract the files.

You can perform this task any time after you have installed Test and you plan to run an API Suite that meets any of the following conditions:

- The API Suite uses transports and the transports require third-party application `Jar` files to be available at the test run time.
- The API Suite in the project has a results database configured.

You can copy the JDK version `Jar` files that you have used to create the JUnit tests.

You can get help on the kubectl commands by running the command: kubectl cp --help from the `/kube` directory. For more information, refer to the kubectl documentation.

1. Use the following table to find the name of the folder that corresponds to the specific third-party application for the transport used in the API Suite.

   Table 590. Name of the folder for the application

Application	Name of the folder to use
Camel	Camel
CentraSite	CentraSite
CICS	CICS
Coherence	Coherence
Database	JDBC
IMS	IMS
Integra	Integra
JMS	JMS
SAP RFC	SAP
Software AG Universal Messaging	SoftwareAGUM
TIBCO EMS	TIBCO
TIBCO Rendezvous	
TIBCO SmartSockets	
WebSphere Application Server Service Integration Bus (SiBus)	WAS

Application	Name of the folder to use
WebLogic	WebLogicJMX
Software AG webMethods	webMethods

2. Open the command prompt from the `/kube` directory and copy the `Jar` files to the folder that corresponds to the application by using the following command:

```
kubectl cp <compressed_files> devops-system/{{main}}-userlibs-0:/data/<application_name>/
```

For example, if you are copying the Database `Jar` files, then the name of the folder is *JDBC* and the command to use is:

```
kubectl cp mysql-connector-java.jar devops-system/{{main}}-userlibs-0:/data/JDBC/
```

Note: You must substitute `{main}` with the release name that you used during the installation of Test.

3. Verify whether the `Jar` files are copied by running the following command:

```
kubectl exec {{main}} -userlibs-0 -n devops-system -- ls /data/<folder_created>
```

For example, if you copied the Database `Jar` files, `mysql-connector-java.jar` to the folder *JDBC*, then the command to verify is:

```
kubectl exec {{main}} -userlibs-0 -n devops-system -- ls /data/JDBC
```

The following information is displayed for the `JDBC` folder:

```
mysql-connector-java.jar
```

Note: You must substitute `{main}` with the release name that you used during the installation of Test.

Results

You have successfully copied the third-party application `Jar` files to the folder in Kubernetes.

What to do next

You can configure the API Suite run. See Configuring an API Suite or an API test run on page 1174.

Server extensions

You can run specific tests in Test by enabling the server extensions.

The server extensions are enabled at the time of installation of Test. When the server extensions are enabled, you can perform the following tasks on Test:

- Add repositories that contain the test assets for the HCL AppScan CodeSweep tests, JMeter tests, JUnit tests or Postman collections.
- Configure a test run for the following test types:
    - **HCL AppScan CodeSweep**
    - **JMeter**
    - **JUnit**
    - **Postman**

To learn more about enabling and disabling the server extensions, refer to the following topics:

Disabling server extensions

When you do not want the server extensions in Test to run on the cluster to save server resources, you can disable server extensions.

Before you begin

You must ensure that the server extensions are enabled at the time of server installation by running the following command:

```
kubectl get pods –n devops-system
```

The status of the server extensions that are enabled is displayed as `Running`.

About this task

You must substitute `{{main}}` with the release name that you provided during the installation of Test in the following procedure.

The `devops-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of `devops-system` in all the instances in this procedure.

1. Log in to the platform that you used to install Test.

   For example, you can log in to the Ubuntu server using an SSH session if you installed Test on Ubuntu.
2. Run the following commands that depend on the server extension you want to disable:

To disable the server extension	Run the command

CodeSweep	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestCodesweepEnabled=false`

To disable the server extension	Run the command

JMeter	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestJMeterEnabled=false`

JUnit	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestJUnitEnabled=false`

Postman	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestPostmanEnabled=false`

All server extensions	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestPostmanEnabled=false \`
	`--set global.hclOneTestJMeterEnabled=false \`
	`--set global.hclOneTestJUnitEnabled=false \`
	`--set global.hclOneTestCodesweepEnabled=false`

3. Run the following command to verify the status of the extensions:

```
kubectl get pods -n devops-system
```

Note: The pod for the server extension that you disabled is not displayed.

Results

You have disabled the required server extensions on Test.

Enabling server extensions

If you find that the server extension is disabled in Test and you want to run the tests supported by the server extension, then you must enable them.

Before you begin

You must ensure that the server extensions are enabled at the time of server installation by running the following command:

```
kubectl get pods -n devops-system
```

The status of the server extensions that are enabled is displayed as `Running`. If the server extension you want to use is not displayed as running, then you must enable the server extension.

About this task

You must substitute `{{main}}` with the release name that you provided during the installation of Test in the following procedure.

The `devops-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of `devops-system` in all the instances in this procedure.

1. Log in to the platform that you used to install Test.

   For example, you can log in to the Ubuntu server using an SSH session if you installed Test on Ubuntu.

2. Run the following commands that depend on the server extension you want to enable:

To enable the server extension	Run the command

CodeSweep	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestCodesweepEnabled=true`

JMeter	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestJMeterEnabled=true`

JUnit	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestJUnitEnabled=true`

Postman	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestPostmanEnabled=true`

All server extensions	`helm upgrade {{main}} -n devops-system ./hcl-onetest-server \`
	`--reuse-values \`
	`--set global.hclOneTestPostmanEnabled=true \`
	`--set global.hclOneTestJMeterEnabled=true \`
	`--set global.hclOneTestJUnitEnabled=true \`
	`--set global.hclOneTestCodesweepEnabled=true`

3. Run the following command to verify a specific extension or all the extensions are enabled:

   ```
   kubectl get pods –n devops-system
   ```

   Result

   The status of the specific extension or all the extensions in the pod is displayed as `Running`.

Results

You have enabled the required server extensions on Test.

Related information

Configuring a run of a JMeter test on page 1377

Configuring a run of a JUnit test on page 1386

Configuring a run of a Selenium test on page 1417

Configuring a run of a Postman test on page 1395

Configuration of a scan of an application code on page 1429

Team space administration

You can find information about team spaces including the initial team space on Test along with the tasks that you can perform to administer a team space.

You can find the following information about team spaces:

- Team space overview on page 742
- Creating a team space on page 745
- Tasks in a team space on page 747
- Tasks in projects in a team space on page 750
- Management of licenses in DevOps Test
- Installation of HCLSoftware Local License Server

Team space overview

Team spaces are logical partitions of Test . Team space is an exclusive, separate, and secure space on Test that enables you to share a single installation among multiple teams.

You can find the following information about team spaces:

- Initial team space on page 742
- Team spaces on page 743
- Migration from earlier releases of Test on page 744

Initial team space

After you install Test and log in for the first time, you enter the **Initial team Space**.

The initial team space is the default logical partition that is created on an instance of Test.

As a server administrator, you must consider the following points:

- After you install Test and log in for the first time, the **Projects** page of the initial team space is displayed.
- You must configure the license server for the initial team space.
- Add or remove users in the team space and assign them different roles in a team space.
- You can assign the role of a *Team Space Owner* to any other member in the initial team space.
- Change the type of the initial team space from the default type of *permissive*.

  The following table describes the type that you can assign to a team space:

Type	Description
Permissive	Specifies that any licensed user of Test can access a team space and get assigned the default roles of a *Project Owner* and *Member* in the team space.
Restrictive	Specifies that all licensed users of Test must request for access to a team space and must be provided access and assigned roles by the *Team Space Owner*. The **Restrictive** type of team space can be either **Public** or **Private**. A **Public** team space is visible in the team spaces dashboard, while a **Private** team space is not visible in the team spaces dashboard.

- You can evaluate the requirement and create team spaces other than the initial team space.

As a licensed user, you must consider the following points:

- When you log in for the first time immediately after the installation of Test, the **Projects** page of the initial team space is displayed.
- After you log in to the initial team space, you are assigned the default roles of *Project Creator* and *Member* of the initial team space.
- On subsequent logins, the **My Projects** page in the initial team space is displayed as the default page.

Team spaces

You can either work with the initial team space or you can create multiple team spaces on Test. The team space facilitates users of the same team space to work together.

Apart from being a container for projects, a team space facilitates the registration of Resource Monitoring agents, designing the system model, and viewing the registered agents, or intercepts.

After you create a team space as a **server administrator**, you must consider the following actions:

Note: You are assigned the default role of a *Team Space Owner* for team spaces that you create.

- Add users in the team space and assign them different roles in a team space.
- Assign the role of a *Team Space Owner* to other team space members, if you want other members to own the team space.

Note: As a *Team Space Owner*, you can remove any member of your team space from the assigned role. The member who is removed from a team space, can send a request and become a member of that team space after the request is accepted.

- If the *Team Space Owner* is no longer a member of a team space, then you as the server administrator can access that team space with the default role of a *Team Space Owner*. You can then assign the role of a *Team Space Owner* to any other member.
- Delete the initial team space or any other team space that you created, if required.

Note: If you delete the initial team space, the users of the initial team space are redirected to the **Team Space Dashboard** page.

You as a server administrator or the *Team Space Owner* must perform the following tasks in a team space before other members can start using the team space:

- Configure the license server.
- Configure the team space type as either *Restrictive* or *Permissive*.
- Configure the SMTP server and add members who can receive notifications about the operations performed in the team space.

As a member of a team space, you must consider the following points:

- When you log in for the first time immediately after the installation of Test, the **Projects** page of the team space is displayed.
- After you log in to the team space, you are assigned the default roles of a *Project Creator*, and *Member* of the team space.
- You can request to become a member of any other team space. After your request is approved, you are assigned the default role of a *Member* of that team space.
- You can become a member of either a **Permissive** or **Restrictive** type of team space.
- On subsequent logins, the **Team Space Dashboard** page is displayed as the default page.

Migration from earlier releases of Test

The initial team space is the only logical partition in Test 10.1.3. If you migrate from an earlier release of Test to 10.2 or later, then all the assets and resources that you created and associated with the initial team space are migrated to associate with the initial team space in Test 10.2 or later.

Note:

The results of migration are as follows:

- All projects are contained in the initial team space.
- If you created a repository in a team space, then you can find the repository in the initial team space in 10.2.
- All agents, and intercepts that you registered with the server are then registered with the initial team space and you can view from the **Agents and Intercepts** page of the initial team space.
- If you configured a resource monitoring agent in any project, then the details of the configured agent are available in the project.
- You can find the details of SMTP configuration on the server. Only you as a server administrator can view the details of SMTP configuration.
- The license server that is configured for the server for earlier release of Test is then configured for the initial team space.
- All users are assigned the role of a *Project Creator* and *Architect* of the initial team space.
- All users with server administrator roles are assigned the default role of a *Team Space Owner* of the initial team space.

Notes:

- All users of earlier releases are migrated to Test 10.2 or later. You can view the list of all migrated users in the application only when they log into the application at least once.
- After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to Test so that other members of the project can access the test assets in that project.

Related information

Team space administration on page 742

Default user administration

Creating a team space

To manage projects in the logical containers of Test , you must create team spaces.

Before you begin

You must have been assigned the role of a server administrator of Test.

About this task

As a server administrator, you can create multiple team spaces on Test.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane, and then click **Create Team Space**.
   Result
   The **New Team Space** dialog is displayed.

3. Provide a name to the team space.
   Result
   The **URL alias** field is automatically filled.

   Note: If you modify the name of a team space, the **URL alias** is also modified automatically.

4. **Optional:** Modify the **URL alias** for the team space, if required.

   Note: You must enter the **URL alias** that starts with a letter.

5. Select an option to set the permissions to access the team space. You can select any one of the following options:
   - **Permissive** – When a team space is set as **Permissive**, the new user gets the permissions automatically on sending the request to become a member of that team space.
   - **Restrictive** – When a team space is set as **Restrictive**, the *Team Space Owner* provides permission to the new user who becomes a member of that team space.

     You can set the visibility mode of the **Restrictive** type team space as **Public** or **Private**.
     - The **Public** team space is visible to all users in the **Team Space Dashboard**.
     - The **Private** team space is visible only to the members of that team space in the **Team Space Dashboard**.

     Note: If you select the **Restrictive** type team space, then the default setting of the team space becomes **Public**.

6. Provide a brief description of the team space.
7. Select the owner of the team space from the list of users who accessed the application.

   The selected user is assigned the role of a *Team Space Owner*.

   Note: You can assign the *Team Space Owner* role to only one user when you create a team space.

8. Click **Create**.
   Result
   The dashboard of the team space is displayed.

Results

You have created a team space.

If you are assigned the role of a *Team Space Owner*, then you can view the name of the team space that you created in the navigation pane.

If you assigned any other licensed user with the role of a *Team Space Owner*, then you can view the team space under the **Other Team Spaces** panel.

What to do next

You must configure the license for your team space. See Configuring licenses for team spaces.

Tasks in a team space

You can find information about the tasks that you can perform as a *Member*, *Project Owner*, *Architect*, or *Team Space Owner* in a team space on Test .

The roles assigned in a team space for users are as follows:

Persona	**Initial team space**		**Any team space**	
	Role	**Notes**	**Role**	**Notes**
Server administrator	*Team Space Owner*	Default role assigned	*Team Space Owner*	Default role assigned
Licensed user	*Team Space Owner*	Role assigned by Server administrator	*Team Space Owner*	Role assigned by Server administrator
	Architect	Role assigned by *Team Space Owner*	*Architect*	Role assigned by *Team Space Owner*
	Project Creator	Default role assigned	*Project Creator*	• Default role assigned in a *Permissive* team space. • Role assigned by *Team Space Owner* in a *Restrictive* team space.
	Member	Default role assigned	*Member*	Default role assigned in either a *Permissive* or *Restrictive* team space.

The following table shows the tasks that can be performed by licensed users with their assigned roles in a team space:

Tasks	**Team Space Owner**	**Architect**	**Project Creator**	**Member**
Configuring licenses	✔			
Configuring a Simple Mail Transfer Protocol (SMTP) server to send email notifications.	✔			
Changing the license server configuration	✔			

Tasks	Team Space Owner	Architect	Project Creator	Member
Adding members in a team space	✔			
Assigning different roles to other team space members	✔			
Modifying configuration details of a team space	✔			
Deleting a team space	✔			
Renaming a team space	✔			
Adding or removing the team space repository from a team space	✔	✔		
Designing a system model	✔	✔		
Creating projects in a team space	✔		✔	
Adding agents	✔	✔	✔	✔
Adding, modifying, or removing resource monitoring sources	✔			
Connecting resource monitoring agents	✔			
Adding resource monitoring labels	✔			
Working with projects in a team space	✔	✔	✔	✔

The following table provides the links to the information about the tasks that can be performed by users with different roles in a team space:

Tasks	Roles	More information
Configuring a team space license server	*Team Space Owner*	Configuring licenses for team spaces
Configuring the Named Users	*Team Space Owner*	Adding users to the Named Users list
Modifying the configuration of a team space	*Team Space Owner*	Modifying the configuration of a team space on page 1717
Viewing the configuration of a team space	All members of a team space	Viewing the configuration of a team space on page 1719
Reassigning the role of team space members	*Team Space Owner*	Managing members and their roles in a team space on page 1721

Tasks	Roles	More information
Becoming a team space member	• *Team Space Owner* • All users can send a request	Becoming a team space member on page 1720
Viewing team spaces	All members of a team space **Notes:** • The **Permissive** and public **Restrictive** team spaces are visible to all the users. • The private **Restrictive** team spaces are only visible to its members.	Viewing team spaces on page 1714
Deleting a team space	*Team Space Owner*	Deleting a team space on page 1723
Adding a project in a team space	*Project Creator*	Creating a project on page 1728
Viewing existing projects	All members of a team space	Viewing active projects on page 1733
Viewing details about the users or license server	All members of a team space	Viewing license details of the team space
Installing remote agents and configuring them to register with Test	All members of a team space	Management of agents on page 1071
Viewing remote agents, or intercepts that are registered with Test in the initial team space	All members of a team space	• Viewing performance agents that are registered with DevOps Test on page 1074 • Viewing API agents that are registered with DevOps Test on page 1081 • Viewing intercepts that are registered with a team space on DevOps Test on page 1459
Adding resource monitoring sources and agents	*Team Space Owner*	Resource monitoring on page 1563

Tasks	Roles	More information
Adding resource monitoring labels to the sources	*Team Space Owner*	Controlling resource monitoring sources in a schedule on page 1595
Viewing the performance metrics	All members of a team space	Viewing resource monitoring usage data on page 1603
Creating a model of the system under test	*Architect*	Creating a system model on page 1034
Adding a repository to a team space	• *Team Space Owner* • *Architect*	Adding a repository to a team space on page 1031
Viewing an existing system model	All members of a team space	Viewing the system model on page 1055

Related information

Team space administration on page 742

Tasks in projects in a team space

You can find information about the roles that you can assign in a project for members of a team space and the tasks that you can perform in projects in a team space.

Members of a team space with the role of a *Project Creator* can create projects in the team space and become the *Owner* of a project. The owners of projects can add *Members* of the team space to a project and assign them the following roles:

- *Owner*
- *Tester*
- *Viewer*

For the list of actions that members assigned any of the roles can perform, see Managing access to the server projects on page 1735.

All projects are created in a team space. When you open a project, the options that are displayed in the navigation pane are specific to the projects within a team space.

When you are working on a project in a team space and you want to return to the options available to work in the team space, you can click the **Back** icon in the navigation pane.

Note: You must be assigned the role of an *Owner*, a *Tester*, or a *Viewer* of a project.

The following table lists the tasks that members of a team space can perform while working with projects in a team space:

Tasks	More information...
Viewing projects in a team space.	Viewing active projects on page 1733
Adding a project.	Creating a project on page 1728
Adding a repository that contains test assets and resources that you want to run on the server.	Adding repositories to a server project on page 1729
Adding secrets to a classification.	Secrets configuration on page 1732
Encrypting the data set resources.	Managing an encrypted dataset on page 976
Configuring a change management system.	Configuration of a change management on page 1608
Adding members to a project and configuring their roles.	Adding users to a project on page 1732
Selecting the global branch in a project to view the test assets and resources.	Selecting the global branch in a project on page 1748
Adding agents to your project, if you are using remote agents as a location to run certain tests.	Adding a performance agent to a project on page 1076
Reading the considerations that you must take into account before you configure a test run.	Prerequisites to running tests on page 1057
Reading the considerations that you must take into account before you configure a run or start a virtual service.	Prerequisites for running HTTP virtual services on page 1455
Configuring a run of the test assets.	Test run configurations on page 1134
Configuring a run of the virtual service resource.	Configuring a run of a virtual service on page 1462
Viewing the progress of a test run.	Viewing the progress of running test assets on page 1444
Managing a running test.	Management of running tests on page 1440
Viewing the results of a test run after the run is completed.	Test results on page 1499
Reviewing the test results and creating defects for test runs from the **Results** page.	Creating Jira issues for the test results on page 1612

Management of resources

You can find information about managing resources on Test along with the tasks that you can perform to define retention rules and monitor resource usage for test results.

By defining the retention rules, you can prevent the system from running out of resources and ensure only the required data is maintained.

Managing retention rules

You can create, edit, and delete retention rules to automatically manage and clean up outdated or no longer needed test result data. This ensures that outdated or unnecessary results are managed and storage is optimized.

Before you begin

You must have been assigned the Team Space Owner role in Test.

About this task

Retention rules help teams manage storage effectively, especially in large-scale testing environments with frequent or long-running test runs. The retention rule provides an automated way to identify and reclaim resources associated with test results. You can choose to perform the following tasks:

- Delete the entire result, removing it from the system completely
- Delete only the test log and report data, reducing disk usage while keeping result metadata

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Manage > Resources**.
   The **Resources** page is displayed.
3. Click the **Rules** tab.

   The **Rules** page displays the following retention rules for the team space:

   - **Delete Stale Branches**: Deletes test results created on Git branches that no longer exist and are older than 3 days. For example, results from branch edits or Git cleanup.
   - **Delete Old Stub Results**: Deletes results of type APISTUB that are older than 30 days.

   Note: You can disable these default rules if required.

   In the **Rules** tab, you can perform the actions that are listed in the following table:

Task	Action
Enabling or Disabling default retention rule	Perform the following steps: a. Click the **Enabled** button to enable or disable the rules. The **Enable Retention Rule** dialog is displayed. b. Click **Ok** to confirm your action.

Task	Action
Adding retention rule	Perform the following steps: a. Click **Add Rule**. The **New Retention Rule** dialog is displayed. b. Enter a name in the **Rule Name** field. c. Select the conditions from the drop-down list. The condition types are: ▪ **Result age is older than (days)** ▪ **Branch status is** ▪ **Result verdict is one of** ▪ **Result size is greater than (MBs)** ▪ **Result labels contains one of** ▪ **Result labels does NOT contain any of** ▪ **The project is one of** ▪ **The test type is one of** ▪ **Max results per unique test** ▪ **Max results per unique test and verdict** **Note:** You can add multiple conditions to define a retention rule. For example, you can select the condition **The project is one of** and choose a project. You can add another condition **Result verdict is one of** and select the value **Pass**. The **Max results per unique test** and **Max results per unique test and verdict** conditions are not displayed if you have selected any of the following conditions: ▪ **Result age is older than (days)** ▪ **Result verdict is one of** ▪ **Result size is greater than (MBs)** The estimated impact is displayed at the bottom of the window, showing how much data will be freed. d. Select one of the following options:

Task	Action
	▪ **Entire Result**: Removes all traces of the test result. ▪ **Just report/testlog**: Retains metadata such as verdict and stop reason. e. Click **Create** to add a rule.
Editing retention rule	Perform the following steps: a. Click the **Edit** icon next to the rule that you want to modify. The **Edit Rule** dialog is displayed. b. Modify the name in the **Rule Name** field. c. Select the conditions from the drop-down list. The conditions types are: ▪ **Result age is older than (days)** ▪ **Branch status is** ▪ **Result verdict is one of** ▪ **Result size is greater than (MBs)** ▪ **Result labels contains one of** ▪ **Result labels does NOT contain any of** ▪ **The project is one of** ▪ **The test type is one of** ▪ **Max results per unique test** ▪ **Max results per unique test and verdict** **Note:** You can add multiple conditions to define a retention rule. For example, you can select the condition **The project is one of** and choose a project. You can add another condition **Result verdict is one of** and select the value **Pass**. The **Max results per unique test** and **Max results per unique test and verdict** conditions are not displayed if you have selected any of the following conditions: ▪ **Result age is older than (days)** ▪ **Result verdict is one of** ▪ **Result size is greater than (MBs)**

Task	Action
	The estimated impact is displayed at the bottom of the window, showing how much data will be freed. d. Select **Entire Result** to remove all the traces of test result or select **Just report/testlog** to retain the metadata (For example: verdict, stop reason). e. Click **Update** to apply the changes.
Running retention rule	Perform the following steps: a. Click the **Execute retention rule now** icon to run the rule immediately to identify and delete results matching its conditions. The **Confirm Recycle** dialog is displayed. b. Click **Yes** to confirm the action
Deleting retention rule	Perform the following steps: a. Click the **Delete Rule** icon to remove the rule from the team space entirely. The **Delete Rule** dialog is displayed. b. Click **Delete** to confirm the action.

Results

You have successfully created, updated, executed, or deleted retention rules.

Viewing resource usage statistics

You can view statistics about the resource usage to monitor and optimize the retention of results.

Before you begin

You must have been assigned the Team Space Owner role in Test.

About this task

When the server runs low on resources, it is difficult to identify which projects are consuming the most resources. This information is accessible to admins at the team space level to make informed decisions about the projects that can be pruned.

1. Log in to Test.

   The **Projects** page of the initial team space is displayed.

2. Click **Manage > Resources**.

   The **Resources** page is displayed.

3. Click the **Statistics** tab.

   The **Statistics** table is displayed with the following details.

Statistics details	Description
Project	The name of the project.
Archive status	Shows whether the project is currently active or archived.
Users	The number of users associated with the project.
Results	The size of the results stored for the project (For example: KB, MB).
Last Result	The date and time of the most recent test result.
Last Repository Modification	The date and time when the project repository was last modified.

   You can sort the table by any column by clicking the column headers so that you can quickly identify projects that consume the most resources or have the most recent activity.

Results

You have viewed detailed resource usage statistics for projects in the team space.

Configuration of an MCP server endpoint

Test provides a Model Context Protocol (MCP) server endpoint. The configuration of the MCP endpoint enables AI assistants within Visual Studio Code (VS Code) to securely integrate with your testing environment, read the test data, and perform specific actions that are supported by the tools.

When you configure the MCP endpoint in VS Code, the supported AI extension functions as an MCP client that communicates with the MCP server. This connection allows the AI assistant to list projects and test assets, retrieve test result data and summaries, and run tests directly within the development environment.

You can use the following endpoint to connect to the MCP server:

```
https://<server_hostname>/test/mcp
```

MCP tools available in Test

The following table lists the MCP tools that are available in Test for running tests and retrieving test results.

Tools	Description	Input	Output

testhub__get_projects	Retrieves a list of projects available in Test.	None	List of available projects
testhub__list_tests	Retrieves a list of tests in Test for a specific project and branch.	projectId, branch	List of tests for the specified project and branch
testhub__get_results	Retrieves test result summaries for multiple test runs in Test.	projectId, branch	Test result summaries for multiple test runs
testhub__get_result_by_id	Retrieves the test run result and detailed result data.	projectId, resultId	Test run result and detailed data
testhub__run_test	Runs a test in a specific project.	projectId, assetId, branch	Details of the test run, including execution ID

Configuring an MCP server endpoint in VS Code

You can configure an MCP server endpoint that enables an MCP-compliant client, such as VS Code, to access specific server-side tools. The endpoint offers several tools that provide access to teamspace, project, and test asset details, along with the ability to run tests and retrieve test results.

Before you begin

You must have completed the following tasks:

- Obtained the hostname or IP address of Test.
- Logged in to VS Code.

About this task

While configuring the endpoint, you will be prompted to authenticate the connection by entering the credentials. And when you run tests using the tool, you must provide an offline token for authentication.

1. Press `Ctrl+Shift+P` to open the command palette in VS Code.
2. Type `MCP add` and select the **MCP: Add Server...** option from the list.
3. Select the **HTTP (HTTP or Server-Send-Events)** option.
4. Paste the URL of the Test MCP server in the **Enter Server URL** field in the following format:

   `https://<server_hostname>/test/mcp`

   For example, `https://testhubserver.com/test/mcp`
5. Enter a name to identify the server, or accept the default when prompted to enter the server ID in the **Enter Server ID** field.

   For example, `myTestHub_mcp`.
6. Select one of the following workspace options when prompted:
   - **Global**: Ensures the tools are available from all your workspaces.
   - **Workspace**: Ensures the tools are available from only the current workspace.

An `mcp.json` configuration file is created.

7. Open the `mcp.json` file from the explorer.
8. Perform one of the following steps to connect to the server:
    - Click **Start** that is displayed above the server name in the `.JSON` file.
    - Click **Extensions**, right-click the server name, and then select **Start Server**.

   VS Code displays a notification stating that the MCP server requires authentication.
9. Click **Allow**.

   A second notification is displayed asking whether you want to open an external website.
10. Click **Open**.

    Your default browser opens and displays the Test login page.
11. Sign in with your Test credentials.

    A consent screen is displayed requesting access to Test MCP tools.
12. Click **Yes** to approve and complete the connection.

    After successful authentication, VS Code displays the number of available MCP tools.

Note: The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session will expire and you are prompted to re-authenticate.

Results

You have configured an MCP server endpoint in VS Code.

What to do next

In the VS Code chat window, you can type a prompt that requires data from the server. For example, you can enter `Get the list of projects` or `List the latest failed test runs`. If you enter a prompt such as `Execute the test <test_name>` for running the test, the tool retrieves the test assets details from the server and then displays a prompt to enter the offline token to process the request. You must generate a token, copy it and paste the token in the VS Code chat window for the test run to complete.

Integration of Test with Measure

When you want to export the test results to Measure, you can use the Test plugin to integrate Test with Measure. After you complete the integration, you can run test assets, and then view the metrics on the Measure dashboard.

Prerequisites

You must have completed the following tasks in Measure before proceeding with the tasks to be performed in Test:

1. Added Test integration in Measure. For more information, refer to the Test Hub plugin documentation.
2. Copied the URL that is displayed on the **Integrations** page in Measure and saved it for later use. Here is the URL sample: https://10.0.2.15.nip.io:9443/reporting-consumer/pluginEndpoint/67eaa3c12b5778a2d1332269/onetest/callback.

Steps to perform in Test

You can perform the following tasks when you want to export test results to your dashboard in Measure:

1. Configure a webhook for Measure.
2. Add additional parameters for test runs on page 759.

Adding additional parameters for test runs

When you want to add additional parameters to be included in test results, you can do so by defining the parameters.

Before you begin

You must have completed the following tasks:

- Read and fulfilled the prerequisites that are listed in #unique_1255.
- Configured a webhook for Measure. See #unique_1253.

1. Click **Execution** in the navigation pane.
2. Select the test that you want to run and click in the **Actions** column to open the **Execute test asset** window.
3. Click **Advanced settings**.
4. Enter the following parameters and values in **Additional Configuration Parameters**:

Parameter name	Value
result.property.buildId	Enter the build ID for which you are running the test in the second edit field as the parameter value. For example, 1.13.4.
result.property.buildUrl	Enter the build URL for which you are running the test in the second edit field as the parameter value. For example, `https://10.10.110.112.nip.io/build/tasks/project/build_test123 .`

5. Click **Add** for each parameter.
6. Click **Execute**.

Results

The additional parameter values are captured while your test runs, and the test results are exported to Measure after the test run is complete.

What to do next

You must perform the following tasks to view the test results in the dashboard of Measure:

1. Create a dashboard in the **Insights** section. For more information, refer to Creating dashboards.
2. Add a chart to the newly created dashboard, and when you click **Add charts**, you must select Functional Tests, Unit Tests, API Tests, and Performance Tests from the **Quality** section. For more information, refer to Adding a chart.

Test Author Guide

This guide describes how to create and edit test assets in Test , and then publish test assets to the remote Git repository that you have configured in your project. This guide is intended for testers and test managers.

For an overview of the test creation and management by using Test Editor, click here to watch the video.

Edit branch overview

When you want to create or modify a test resource in Test , you must use an Edit branch. An Edit branch is a temporary branch created from the selected branch in the repository configured within the project, where you can save the created or edited resource.

You must be a *Team Space Owner*, *Project Owner*, or *Member* with a *Tester* role to create or edit a test resource.

When you want to create or edit a test resource, you must first select the branch in the project repository into which you want to add the created resource or the branch that contains the resource you want to edit.

Note: If you create an Edit branch from a read-only repository, a warning is displayed that publishing will fail.

When you create the Edit branch, any project member can use your Edit branch while the branch is available. When you provide a name for the Edit branch, you must adhere to the rules of Git branch naming conventions. For more information, refer to the Git documentation. You can view all the Edit branches from the **Branch** drop-down list when other project members create Edit branches.

Note: If there are too many simultaneous requests for Edit branch creation in a project, Git might take some time to complete the branch creation.

To differentiate the remote branch and Edit branch from the **Branch** drop-down list, an edit icon and the initials of the member who created the Edit branch are shown for the Edit branch. After you create the Edit branch, you can also view the Edit branch by accessing the remote Git repository in a browser.

If you want to save the created or edited resource to the branch in the project repository, publish the changes from the Edit branch to the selected branch. When the project repository is replicated automatically to the remote repository, the created or edited resources are then available in the remote repository in the branch they were published. The Edit branch that you used is automatically deleted after publishing, and it disappears from the **Branch** drop-down list and from the remote Git repository when you perform any of the following actions:

- Publish changes to the remote Git repository
- Discard the changes made in all test assets

Note: After the Edit branch disappears, the **Branch** field is set to the previously selected branch name.

If publishing the changes in edit branch fails due to conflict with remote branch, you can push the edit branch to remote repository by using **Push** option in the **Publish** dialog. Then create a pull request to merge your changes into the target branch by resolving the conflicts manually in Git.

When multiple users use the same edit branch, the modifications, renaming, moving, or deletions made by a user are notified to the other users. The notifications are displayed in the following scenarios:

- A user modifies the test asset. In this scenario, the notification also provides the following options:
  - **Reload**: Refreshes the page with the updated details.
  - **Compare**: Displays the **Compare <asset_name>** pop-up with the last saved version and the current version for comparison. The changes are highlighted.
- A user moves or renames a test asset.
- A user deletes a test asset.
- A user discards the changes on the **Changes** page.

Test Editor overview

You can use the **Test Editor** tab to create and manage test assets such as applications, Web UI tests, test suites, and environments in Test .

You can navigate to the **Test Editor** page and **Suite Editor** page from the **Author > Test Editor** tab in Test.

When you work on individual tests by creating or configuring test assets, the **Test Editor** page streamlines the entire testing process. You can efficiently organize applications, tests, suites, smartshots, environments that you created or modified in your project by using the test navigator panel in **Test Editor**.

Related information

Application management on page 765

Web UI test management on page 769

Test suite management on page 829

Test Editor page on page 763

Suite Editor page on page 764

Test navigator panel

The **Test Editor** page contains a central hub called the test navigator panel. You can use the test navigator panel to manage and organize test assets in multiple folders of your project.

You can right-click a test asset in the Test Navigator panel and select an option from the context menu to perform the following actions:

Actions	Description
Search	You can search for a test asset in a project by entering the name of the test asset in the **Search** field.
New	You can create an application, test, suite, or an environment in a project. See Creating an application on page 766, Recording a Web UI test on page 771, Creating a suite on page 830, and Creating an environment on page 870.
Edit	You can modify application components, Web UI tests, suites, or environments when you want to run a test asset with different values. See Modifying application components on page 767, Editing a Web UI test on page 793, Modifying a suite on page 831, and Modifying an environment on page 872.
Rename	You can change the name of an application, test, suite, environment, or smartshot within a project.
Move	You can change the location of a test, suite, environment, or smartshot and move it under a different application within a project. When moving a single asset, you can choose to include dependencies that are referenced only by the selected asset. If a dependency asset is shared by another asset, you cannot select that dependent asset for moving.
Delete	You can delete an application, test, suite, environment, or smartshot when it is no longer required in a project. When deleting a single asset, you can choose to include dependencies that are referenced only by the selected asset. If a dependency asset is shared by another asset, you cannot select that dependent asset for deletion. See Deleting a test asset on page 827, Deleting a suite on page 845.
Compare	You can view and compare changes made in the edit branch with unpublished changes in the Web UI test, suite or environment. You can select the **Compare** option, which appears when you right-click the test asset.
Save As	You can create a copy of an existing application, test, suite, or an environment with a new name in a project. You can select the **Save As** option, which appears when you right-click the test asset.
Collapse All	You can minimize all expanded test assets in the project view to simplify navigation and focus on top-level items.
Expand All	You can expand all sections or folders in the project view to quickly view the all the test assets.

You can select multiple test assets in the **Test Navigator** by pressing `Ctrl` and clicking individual assets, or by pressing `Shift` and clicking to select a range of assets.

You can use the following options in the test navigator toolbar when you manage and organize tests, applications, and suites across multiple folders of your project.

Icon	Description
	You can use the **Virtual Folder** icon to view folders at different levels. The available options are as follows: • **Root Folders**: Shows virtual folders at the top, followed by the folder structure of the repository. • **Embedded Folders**: Shows the folder structure of the repository first, with files organized into virtual folders. • **None**: Excludes virtual folders entirely, and presents only the original folder structure of the repository.
	You can use the **Link with editor** icon to connect Test Editor tabs to their corresponding locations in the test navigator panel. By using this icon, you can easily navigate and quickly access specific test assets within your project.
	You can apply a filter to view folders by their asset type. You can select one or more of the following asset types and apply the filter: • Datasets • Tests • Environments • Others • Smartshots • Suites
	You can use the **Save All** icon to save all the changes made to multiple test assets or folders within your project.

Test Editor page

The **Test Editor** page consists of two sections: a left pane and a right pane. You can use the left pane to view, edit, and organize the test. You can use the right pane to manage and analyze the test within the project.

You can use the following options in the left pane of the **Test Editor** page:

Options	Actions
View	In the default setup, the test is displayed in the **Visual** view. To view the test in **Code** view, you must click the **Code** option. The view that is selected will be active till you close the test.
Find Step	Enter the name of the test in the **Find Step** to search the test assets in the current test.
	Select the icon to begin the test run.

Options	Actions
Undo	Select the **Undo** icon to undo the recent changes.
Redo	Select the **Redo** icon to redo the recent changes.
Save	Select the **Save** icon to save all the changes made to the test.
Publish	Select the **Publish** icon to publish the test.

You can use the following tabs in the right pane of the **Test Editor** page:

Tabs	Actions
Catalog	Select the **Catalog** tab to add additional steps to the tests. See Adding user actions in a test from smartshots on page 796, New test steps from the General catalog on page 798, and New steps for UI-related actions and user actions in a test from the UI catalog on page 806.
Results	Select the **Results** tab to view the test results. See Running a Web UI test on page 785.
Usage	Select the **Usage** tab to see where the current test is referenced in other tests. You can click the icon to open the referenced tests. If there are no references to the current test, then the tab displays that the current test has no dependants.
Details	Select the **Details** tab to view the Git details such as **Asset ID**, **Repo**, **Path**, **Branch**, and **Commit**.

Related information

Web UI test management on page 769

Suite Editor page

The **Suite Editor** page consists of two sections: a left pane and a right pane. You can use the left pane to view, edit, and organize the suite. You can use the right pane to manage and analyze the suite within the project.

You can use the following options in the left pane of the **Suite Editor** page:

Options	Actions
View	In the default setup, the suite is displayed in the **Visual** view. To view the suite in **Code** view, click the **Code** option. The view that is selected will be active till you close the test.

Options	Actions
Find Step	Enter the name of the suites and tests in the **Find Step** to search suites and tests in the current suite.
	Select the icon to begin the test run.
Undo	Select the **Undo** icon to undo the recent changes.
Redo	Select the **Redo** icon to redo the recent changes.
Save	Select the **Save** icon to save all the changes made to the suite.
Publish	Select the **Publish** icon to publish the suite.

You can use the following tabs in the right pane of the **Suite Editor** page:

Tabs	Actions
Results	Select the **Results** tab to view the test results. See Running a suite sequentially on page 834 and Running a suite concurrently on page 835.
Usage	Select the **Usage** tab to see where the current suite is referenced in other suites. You can click the icon to open the referenced suites. If there are no references to the current suite, then the tab displays that the current suite has no dependants.
Details	Select the **Details** tab to view the Git details such as **Asset ID**, **Repo**, **Path**, **Branch**, and **Commit**.

Related information

Test suite management on page 829

Application management

When you want to create and manage Web UI tests in Test , you must first create an application within the project.

By creating an application in the project, you can streamline test management by ensuring that the tests are run on the correct application. The application organizes the process by linking each test asset to a specific application. Thereby improving the efficiency of test run.

Components of an application

The components are the key elements in the application that you might want to test. These components include the following details:

- **URL**: The web address where the application is hosted.
- **Description**: A brief explanation or details about the application.
- **Image**: A graphic representation or a logo of the application.

Managing applications primarily consists of two key activities:

- Creating an application on page 766.
- Modifying application components on page 767.

Creating an application

You can create an application within your project in Test and select it to create or record a Web UI test for that specific application.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

When you create an application within your project, you can initiate the test creation process, capture interactions and behaviors on the Web UI, and automate testing workflows.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Navigate to **Author > Test Editor**.
   The **Test Editor** page and test navigator panel are displayed.
4. Perform any of the following actions:
   - Click the **Action Menu** icon and select **Application**.
   - Right-click the project repository in the test navigator panel and select **New Application**.
   - Click **Create Application** on the **Test Editor** page, which is displayed when no applications are created in your project.

The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 6 on page 767.

5. Select your action to either select an existing Edit branch or create an Edit branch:
    - Perform the following steps to select an existing Edit branch:
        a. Select **Use an existing edit branch**.
        b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

        c. Click **Ok**.
    - Perform the following steps to create the Edit branch:
        a. Select **Create new edit branch**.
        b. Enter a name for the Edit branch in the **Branch Name** field.
        c. Click **Save**.
6. Type a unique name in the **Name** field of the **Create Application** dialog.
7. Click **Create**.
   The application is successfully created and it opens in the **Test Editor** for further configuration.
8. Type the web address in the **URL** field.
   Test detects and displays the logo of the website in the **Image** field.
9. Type a brief summary or relevant details about the application in the **Description** field.
10. Click the **Save** icon to save the details.

Results

You have created an application.

What to do next

You can perform the following tasks:

- Record a Web UI test. See Recording a Web UI test on page 771.
- Create a Web UI test. See Creating a Web UI test by using the empty test option on page 777.
- Modify application components. See Modifying application components on page 767.

Modifying application components

You can modify the existing application components within your project in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that your project contains an application.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Navigate to **Author > Test Editor**.
   Result
   The **Test Editor** page and test navigator panel are displayed.
4. Select the **Application** in the test navigator panel and click **Edit**.
   Alternatively, right-click the application in the test navigator panel and click **Edit**.
   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 6 on page 768.
5. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.
6. You can modify the following details of an application in the **Test Editor** page:

- **URL:** Update the web address if the location of the application has changed.
- **Description:** Modify the summary of the application or details to reflect any new information or changes to the application.
- **Image:** Change the logo or graphical representation of the application by clicking the **Edit Icon** in the **Image** field.

7. Click the **Save** icon to save the details.

Results

You have modified the application components.

Web UI test management

You can test your web applications by recording Web UI tests in Test . Test supports industry-standard browsers such as Google Chrome, Microsoft Edge, and Mozilla Firefox to record and play back Web UI tests. After the playback, you can evaluate the test results.

You can perform the following tasks to create and then manage your test assets in Test:

- Recording a Web UI test on page 771
- Creating a Web UI test by using the empty test option on page 777
- Running a Web UI test on page 785
- Viewing step-level results and updating tests on page 789
- Editing a Web UI test on page 793
- Adding user actions in a test from smartshots on page 796
- New test steps from the General catalog on page 798
- New steps for UI-related actions and user actions in a test from the UI catalog on page 806
- Adding conditional logic and redefining the flow in tests from the Flow catalog on page 814
- Creating and assigning test variables on page 847
- Assigning an object property to a test variable on page 849
- #unique_1277
- Configuring timeouts for tests and test steps on page 824
- Deleting a test asset on page 827

Web UI test creation

You can create Web UI tests by using the recording option or the empty test option in the Test . Test automatically enables web browsers and provides DevOps Test Runtime or the recording agent to configure the test environment before you start recording tests for web applications.

You can create tests for the web applications that you have already configured in **Test Editor**. When you create a test, a `.dtx.yaml` file is created and listed in the test navigator panel. Test captures mouse and keyboard actions that the user

performs to interact with the web pages when you record a test. Test supports the following HTML5 controls on the web pages:

- Text box
- Password field
- Text area
- Button
- Checkbox
- Drop-down list
- Date selector
- Radio button
- Slider
- JavaScript popup boxes such as alert, confirm, or prompt dialogs
- Drag-and-drop boxes
- File upload field

Test also supports the use of multiple tabs and windows to record a test.

Application URL management

If you have configured the application with a URL, then Test opens the browser with the configured URL when you record a test for that application. Otherwise, you must enter or paste the URL in the browser when you record a test. When the application contains a URL, Test creates a variable App_<app_name>_URL to store the value of the URL. Otherwise, if you manually enter or paste the URL, then the URL is embedded in the **Open** step.

When you record a test for an application that requires credentials to be supplied in an authorization dialog, before running the test, you must edit the test to modify the URL by adding the credentials as follows:

https://username:password@url

For example, https://test_user:Mypassword!23@sample.com

Also, you must convert any special characters in the username and password by using the URL encoding format.

For example, you can convert Mypassword!23 to Mypassword%2123.

Drag-and-drop boxes

The **Drag** action that is recorded contains two values: From and To. The **From** tab in the **Object Identified By** section displays the identifier of the object that is being dragged, and the **To** tab displays the identifier of the new location for the dragged element.

File upload scenario

You can validate the file upload functionality for a web application by recording and playing back the file upload function. When you record a file upload function, a **Type** step is added with a variable to save the file path and file name. The uploaded file is listed under `blob\Others` in the test navigator panel.

Support for device emulation

You can use the device mode of browsers to record tests, and run tests or suites. Test supports the following device profiles:

Device profile	Default browser
Phone (Small)	Edge
Phone (Medium)	Chrome
Phone (Large)	Edge
Phone (X-Large)	Safari (only on a Mac system)
Tablet (Small)	Edge
Tablet (Medium)	Chrome
Tablet (Large)	Edge
Tablet (X-Large)	Safari (only on a Mac system)

Recording a Web UI test

You can record screens and user actions on the web page of an application to create a Web UI test in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760.
- Ensured that your project contains an application.

About this task

When you choose to create a Web UI test by using the recording option, the installation status of the recording agent is displayed. Initially, when you are recording a test for the first time, the status is displayed as 'Not installed' with the **Download Installer** link to install the recorder agent. The `devops-test-runtime.msi` file is downloaded. You must then double-click the .msi file to install the agent. After the installation is complete, you must click **Verify** to continue with the recording process.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Perform one of the following options:

- Click **Create asset > Create Test** on the **Test Editor** page.
- Click the **Action menu** icon in the test navigator panel to open the menu and click the **Test** option.
- Right-click the application in the test navigator panel, select **New > Test**.

Alternatively, you can create a test by clicking the **Create Test** icon on the application **Details** page if there is an edit branch active.

The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 6 on page 772.

5. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

     Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.
6. **Optional:** Click **Open** in the browser notification that might be displayed for you to confirm the opening of DevOps Test Runtime.

   The browser notification is displayed only when the recording agent is already installed.
7. Perform the following steps in the **Details** tab of the **Create Test** dialog:
   a. Select an application for which you want to create a test from the **Application** drop-down list.

      The application folder is selected as the default location for the test.

   b. **Optional**: Select a folder if there are subfolders in **Location** to save the test assets that you are creating.
   c. Enter a name for the test in **Asset Name**.
8. Click **Next**.

   The **Source** tab is displayed.
9. Select the **By recording** option to record a test in the **Select source** drop-down list.

   Test checks whether the recording agent is installed on your computer.
10. Perform one of the following steps depending on the installation status of the agent:

If...	Then...
The agent is not installed and the screen displays the following options: ◦ Windows (64-bit .msi) ◦ Mac OS (64-bit .zip) ◦ Debian/Ubuntu (64-bit .deb) ◦ RHEL/Fedora/OpenSUSE (64-bit .rpm)	Perform the following steps to install the agent on Windows: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.msi` file is downloaded. b. Double-click the `.msi` file to install the agent. The installation window with the progress bar is displayed. c. Click **Proceed** to accept the download of the proprietary certificate if prompted. d. Click **Verify** to confirm the agent installation. A browser notification might get displayed for you to confirm the opening of the agent. You must click **Open**. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Mac: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.zip` file is downloaded. b. Extract the `.zip` file: unzip devops-test-runtime.zip c. Quarantine the installer: xattr -dr com.apple.quarantine ./* d. Run the installer: ./install.sh e. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Ubuntu: a. Click **Download** to download the recording agent installer. The `devops-test-runtime_11.0.9_amd64.deb` file is downloaded. b. Navigate to the directory where the installer is downloaded: cd <download-directory>

If...	Then...
	c. Run the installer: sudo apt install ./devops-test-runtime_11.0.9_amd64.deb d. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Linux: a. Click **Download** to download the recording agent installer. The `devops-test-runtime-11.0.9-1.x86_64.rpm` file is downloaded. b. Navigate to the directory where the installer is downloaded: cd <download-directory> c. Run the following command: INGRESS_DOMAIN=<domain_name> dnf localinstall ./devops-test-runtime-11.0.9-1.x86_64.rpm d. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
The agent is installed, but not updated, and the screen displays the following options: ◦ Windows (64-bit .msi) ◦ Mac OS (64-bit .zip) ◦ Debian/Ubuntu (64-bit .deb) ◦ RHEL/Fedora/OpenSUSE (64-bit .rpm)	Perform the following steps to update the agent on Windows: a. Click **Upgrade**. The installation screen with the progress bar is displayed. The latest DevOps Test Runtime is installed and configured. b. Click **Proceed** to accept the download of the proprietary certificate if prompted. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Mac: a. Click **Download** to download the recording agent installer. b. Uninstall the existing agent: rm -rf ~/Applications/devops-test-runtime.app c. Extract the downloaded `.zip` file:

If...	Then...
	unzip devops-test-runtime.zip d. Quarantine the installer: xattr -dr com.apple.quarantine ./* e. Run the installer: ./install.sh f. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Ubuntu: a. Click **Download** to download the recording agent installer. b. Uninstall the existing agent: sudo apt remove devops-test-runtime c. Navigate to the directory where the installer is downloaded: cd <download-directory> d. Run the following command: sudo apt install ./devops-test-runtime_11.0.9_amd64.deb e. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
	Perform the following steps to install the agent on Linux: a. Click **Download** to download the recording agent installer. b. Uninstall the existing agent: sudo yum remove devops-test-runtime c. Navigate to the directory where the installer is downloaded: cd <download-directory> d. Run the installer: sudo INGRESS_DOMAIN=<domain_name> dnf localinstall ./devops-test-runtime-11.0.9-1.x86_64.rpm

If...	Then...
	e. Click **Verify** to confirm the agent installation. The additional options for recording are displayed. Go to step 11 on page 776.
The agent is installed.	Go to step 11 on page 776.

11. Select a browser from the **Browser** drop-down list.
    The following browsers are listed:

Operating system	Browsers
Windows, RHEL, or Ubuntu	Chrome, Edge, and Firefox
Mac OS	Chrome, Edge, Firefox, and Safari

    Also, the following device profiles are listed:

Device profile	Default browser
Phone (Small)	Edge
Phone (Medium)	Chrome
Phone (Large)	Edge
Phone (X-Large)	Safari (only on a Mac system)
Tablet (Small)	Edge
Tablet (Medium)	Chrome
Tablet (Large)	Edge
Tablet (X-Large)	Safari (only on a Mac system)

    Note: If you selected the Edge or Firefox browser on a Mac system, you might see the following error: `The recorder encountered a problem: No compatible WebDriver was found or the browser is already running.`

    To resolve this issue, you must check whether the `~/.cache/selenium` folder is password protected, and then change the access permission to make it a regular folder.

12. Select **UI** from the **Interactions** drop-down list.
13. Click **Record**.
    A new browser window is displayed, and the application that you selected for recording opens. If you selected Chrome, Edge, or Firefox, the browser opens in a temporary profile with no cache or cookies. For Safari, when the browser opens, you must manually clear the cache and cookies as required.

Note: If you do not want to continue with the application that is open, you can type or paste the required URL to record the test. Test inserts the **Open** step to capture the new URL that you used for recording the test.

14. Click the UI controls and perform user actions in the application to record test steps.
    You must allow some delay of a few seconds between two user actions to ensure that the UI elements are recognized and registered properly.
15. Close the browser to stop the recording.
    The '`The asset is now ready for your review`' message is displayed in the **Create Asset** dialog.
16. Click **Done**.
    The newly recorded test is displayed in **Test Editor**. The test is displayed in the **Visual** view, which is the default view. You can switch to the **Code** view to view all the step details.

Results

You have recorded a Web UI test.

What to do next

You can perform the following tasks:

- Run the test and view the results. See Running a Web UI test on page 785.
- Edit the test in Test Editor. See Editing a Web UI test on page 793.
- Create variable data for the test. See Test variables on page 846.

Related information

Failed to install DevOps Test Runtime on Windows 11 25H2 on page 1796

Failed to load the extension in Google Chrome while recording on page 1797

Creating a Web UI test by using the empty test option

You can create a Web UI test by selecting the empty test option and then adding the steps and elements either through the catalog in the Visual view or through the written code in the Code view.

Before you begin

You must have completed the following tasks:

- Ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760.
- Ensured that your project contains an application.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Perform one of the following options:
   - Click **Create asset > Create Test** on the **Test Editor** page.
   - Click the **Action menu** icon in the test navigator panel to open the menu and click the **Test** option.
   - Right-click the application in the test navigator panel and select **New > Test**.

   Alternatively, you can create a test by clicking the **Create Test** icon on the application **Details** page if there is an edit branch active.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 6 on page 778.
5. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

     Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.
6. **Optional:** Click **Open** in the browser notification that might be displayed for you to confirm the opening of DevOps Test Runtime.
   The browser notification is displayed only when the recording agent is already installed.
7. Perform the following steps in the **Details** tab of the **Create Test** dialog:
   a. Select an application for which you want to create a test from the **Application** drop-down list.

      The application folder is selected as the default location for the test.

   b. **Optional**: Select a folder if there are subfolders in **Location** to save the test assets that you are creating.
   c. Enter a name for the test in **Asset Name**.
8. Click **Next**.
   The **Source** tab is displayed.

9. Select the **Create an empty test** option in the **Select source** drop-down list.
10. Click **Done**.
    An empty test is displayed in **Test Editor**. The test is displayed in the **Visual** view on the **Test Editor** page. You can also switch to the **Code** view.
11. Choose one of the following options:
    - Add your code to build the test in the **Code** view. Go to step 19 on page 779.
    - Add steps in the **Visual** view. Go to step 12 on page 779.

    Note: You can also copy the steps from an existing test and paste them in the empty test by right-clicking the **Configure settings and data** step.

12. Drag and drop an element from one of the following catalogs to the **Test Editor** page:
    - **General**
    - **UI**
    - **Flow**

    A test step is added.
13. Click the step to add the details.
    The step details pane is displayed and the Smartshot pane does not display any image.
14. Drag and drop a `.jpeg` file to the Smartshots pane from the **shot > Smartshots** folder under the selected application.
15. Select an option from the **UI object** drop-down list.
16. Select a user action in the **Action** drop-down list or enter a value in the **Value** field depending on the type of step you have added from the catalog.
17. Click the **click here to add identifier** link and then in the **Object Identified By** section, specify the condition and value that identifies the UI object.

    Alternatively, you can drag-drop a property that identifies the object from the **Properties** list.
18. **Optional:** Click the **Click here to add relative** to link to define the location and name of the related object.
19. Click the **Save** icon to save the modifications.

Results

You have created a Web UI test by using the empty test option.

What to do next

You can perform the following tasks:

- Run the test and view the results. See Running a Web UI test on page 785.
- Edit the test in Test Editor. See Editing a Web UI test on page 793.
- Create variable data for the test. See Test variables on page 846.

Related information

New test steps from the General catalog on page 798

New steps for UI-related actions and user actions in a test from the UI catalog on page 806

Adding conditional logic and redefining the flow in tests from the Flow catalog on page 814

Test creation with Test Genie

Test Genie is an AI-powered capability in Test that automatically translates your natural language instructions into usable test scripts for web applications.

Traditionally, creating automated tests requires manually recording interactions against a live application or writing a test code. Test Genie accelerates this process so that you can define test steps by using plain text prompts. You provide a descriptive prompt, such as adding specific items to a shopping cart and proceeding to checkout, and the AI generates the corresponding test assets.

To improve the accuracy of the generated test steps, Test Genie uses the application context. When you initiate a generation request, the system passes your natural language steps along with the application URL and any application descriptions to the AI model. This context helps the AI understand the target interface and correctly map your instructions to the underlying web elements.

Supported AI providers

Disclaimer:

This release supports Anthropic in Test as a Tech Preview. The Tech Preview is intended for you to view the capability of Anthropic for test generation offered by HCL Test, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

To use Test Genie, you must configure at least one AI provider in the **Infrastructure > Cloud Credentials** page of Test. Test Genie supports the following AI providers:

- Anthropic - Tech Preview
- OpenAI

For optimal results, you can use OpenAI with the `GPT-5-mini` model.

Capabilities and limitations

Test Genie streamlines the initial creation of test scripts, but it operates within specific boundaries regarding test validation and test modification. The following points describe the scope of Test Genie:

- **Generative scope:** Test Genie generates the navigational and transactional steps of a test, such as clicking buttons, entering text, and navigating between pages. It also provides test generation details, including the model used and token usage metrics.
- **Iterative modifications:** You can use AI prompts to modify or append steps to a test that Test Genie has already generated. To modify the AI-generated steps, you must submit a revised prompt and then generate the test again.
- **Verification points:** The AI does not automatically generate or insert verification points into the test script. You must manually add verification points to the generated steps to validate application behavior and data.
- **Datasets:** Test Genie does not create datasets or automatically link existing dataset files to the generated test. You must configure datasets manually.

Creating a Web UI test with Test Genie

You can use the AI-powered Test Genie to generate test steps for a web application based on your natural language instructions.

Before you begin

You must have completed the following tasks:

- Ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760.
- Ensured that your project contains an application.
- Configured at least one AI provider in the **Cloud Credentials** page. See Adding cloud credentials of AI providers on page 1122.

About this task

Test Genie accelerates test creation by translating your written instructions into a test script. The AI provider uses the application URL and the details in the application description to improve the accuracy of the generated steps.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Perform one of the following options:
   - Click **Create asset > Create Test** on the **Test Editor** page.
   - Click the **Action menu** icon in the test navigator panel to open the menu and click the **Test** option.
   - Right-click the application in the test navigator panel and select **New > Test**.

Alternatively, you can create a test by clicking the **Create Test** icon on the application **Details** page if there is an edit branch active.

The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 6 on page 782.

5. Select an existing edit branch or create a new edit branch.
6. Perform the following steps in the **Details** tab of the **Create Test** dialog:
    a. Select an application for which you want to create a test from the **Application** drop-down list.

       The application folder is selected as the default location for the test.

    b. **Optional**: Select a folder if there are subfolders in **Location** to save the test assets that you are creating.
    c. Enter a name for the test in **Asset Name**.
7. Click **Next**.

   The **Source** tab is displayed.
8. Select the **Create an empty test** option in the **Select source** drop-down list.
9. Enter a set of natural language instructions for your test in the **Steps** text box.

   For example, for testing an e-commerce website, you can enter:

```
Add three sports category items to the cart

Remove one item during the checkout process

Log in with the credentials

Proceed to order
```

10. Click **Done**.

    An empty test is displayed in **Test Editor**.
11. Click the **Test Genie** tab.

    The instructions you entered during the test creation are displayed in the **Steps** text box under the **Test Genie** tab.
12. Review, modify, or add to your natural language instructions in the text box.
13. Click **Generate**.

    The **Generate Test** dialog is displayed.
14. Perform the following steps in the dialog:
    a. Enter or modify the web address of the application under test in the **Start URL** field as required.
    b. Select an AI provider from the configured **AI Provider** list.
15. Click **Generate**.

    The test generation process starts.
16. Click the **Results** tab and then click the **Show in progress page** icon .

    The **Progress** page is displayed. If the page contains many jobs, you must create a filter with the rule **AI Assisted Generation** to filter for AI-assisted jobs.
17. Select **View AI interactions** from the listed actions in the **Actions** column of the AI-assisted job.

    A new tab opens with the console output. You can view the AI interactions and session logs on the progress page by refreshing the view.

18. Switch to the test editor tab after the execution job is complete in the **Progress** page.
    The **Reload** button is displayed with the following warning: `A newer version of this resource is available.`
19. Click **Reload**.
    The AI-generated test steps are displayed.

Results

You have created a Web UI test with Test Genie.

What to do next

You can perform the following steps:

- You can view the execution results, the model used, and the token usage details in the **Test Genie** tab.
- Play back the generated test. See Running a Web UI test on page 785 or Running server test assets on page 1321.

Related information

Modification of Web UI tests on page 792

Viewing the progress of running test assets on page 1444

Modifying a Web UI test created by using Test Genie

You can modify the natural language instructions that you provided to generate a test by using the AI-powered Test Genie. After modifying the instructions, you can regenerate the test to update it.

Before you begin

You must have completed the following tasks:

- Ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760.
- Ensured that you have created a Web UI test by using Test Genie. See Creating a Web UI test with Test Genie on page 781.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.yaml` file to open the test.
   The test steps are displayed on the **Test Editor** page.

5. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 784.
6. Select an existing edit branch or create a new edit branch.
7. Click the **Test Genie** tab.

   The instructions you entered during the test creation are displayed in the **Steps** text box under the **Test Genie** tab.
8. Review, modify, or add to your natural language instructions in the text box.
9. Click **Generate**.

   The **Generate Test** dialog is displayed.
10. Perform the following steps in the dialog:
    a. Enter or modify the web address of the application under test in the **Start URL** field as required.
    b. Modify the AI provider from the configured **AI Provider** list if required.
11. Perform one of the steps as required:
    - If you want to first clear the steps that are already generated, retain the selection in the **Reset script before generation** checkbox.
    - If you want to append the new steps after the last step in the generated test, clear the **Reset script before generation** checkbox.
12. Click **Generate**.

    The test generation process starts.
13. Click the **Results** tab and then click the **Show in progress page** icon .

    The **Progress** page is displayed. If the page contains many jobs, you must create a filter with the rule **AI Assisted Generation** to filter for AI-assisted jobs.
14. Select **View AI interactions** from the listed actions in the **Actions** column of the AI-assisted job.

    A new tab opens with the console output. You can view the AI interactions and session logs on the progress page by refreshing the view.
15. Switch to the test editor tab after the execution job is complete in the **Progress** page.

    The **Reload** button is displayed with the following warning: `A newer version of this resource is available.`
16. Click **Reload**.

    The modified test steps are displayed.

Results

You have modified a test that was created by using Test Genie.

What to do next

You can perform the following steps:

- You can view the execution results, the model used, and the token usage details in the **Test Genie** tab.
- Play back the generated test. See Running a Web UI test on page 785 or Running server test assets on page 1321.

Related information

Modification of Web UI tests on page 792

Viewing the progress of running test assets on page 1444

Running a Web UI test

You can run a Web UI test in Test after you record the test, and to verify that the web application performs as expected.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensured that your project contains a Web UI test in the test navigator panel.

About this task

When you play back the test, the test runs in the browser that is used to record the test. The progress bar shows the progress of the test, while the three color-coded boxes display the counts of passed steps, failed steps, and errors. You can view the step details of the first failed step or error step by clicking the respective color-coded boxes. As the test runs, you can view checkmarks for the steps that pass and crossmarks for the steps that fail. After the test run is complete, you can view the overall results on the **Results** page.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.yaml` file to open the test.
   The test steps are displayed on the **Test Editor** page.
5. Perform the following steps to specify the browser that you must use for the test run:
   a. Click the **Configure settings and data** step.

      The **Configure settings and data** panel is displayed.

   b. Select one of the following options from the **Browser** drop-down list:
      - Browser
      - Device profile
      - Device from the list of devices for the configured device cloud

      The browser that is used to record the test is selected as the default browser and the browser specified here modifies the default browser.

Note: You must have accepted the license agreements for Google Chrome and Microsoft Edge browsers in the **Third-Party Agreements** section after navigating to **Account** > **Account Settings**. If the third-party agreements are not yet accepted, the list options display links so that you can navigate directly to the agreement and accept the terms.

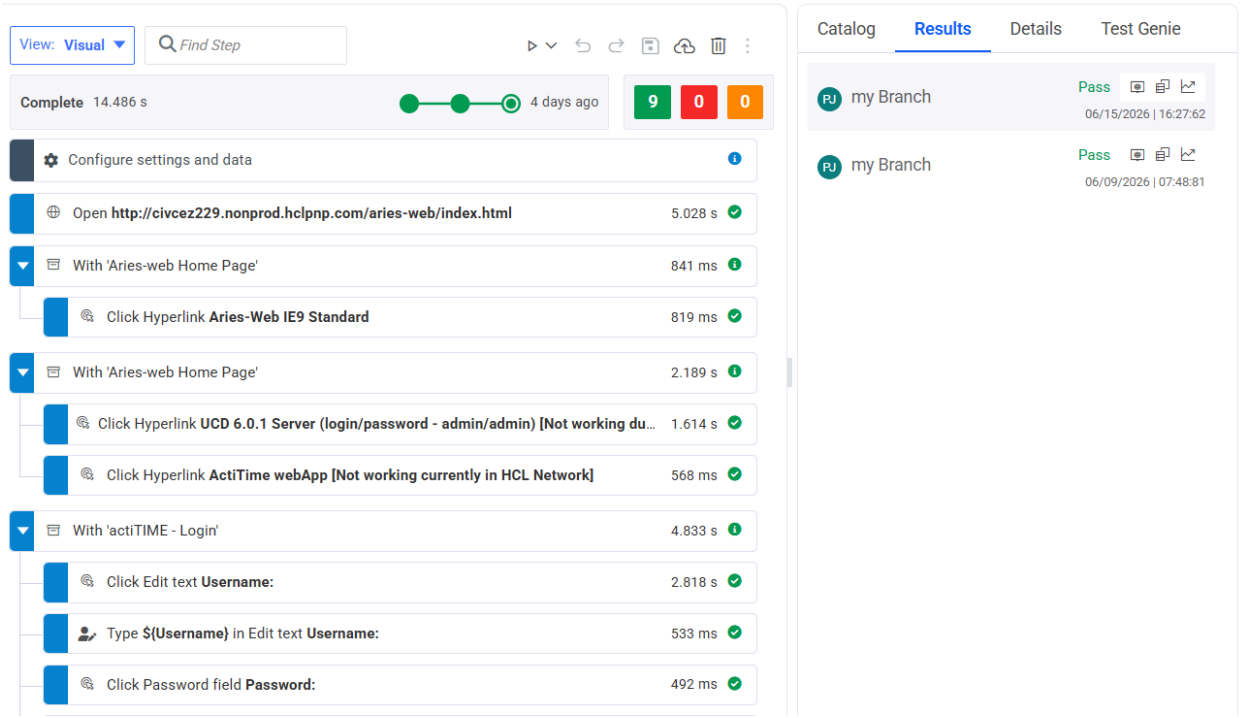
6. Click the icon, and then select the **Run on Default Cluster** option to begin the test run.

   You can also select the **Run on Personal Agent** option if the personal agent is in the ready state.

   You can view the following stages during the test run:
   - **In transition**: The test setup is in progress.
   - **Running**: The test run is active.
   - **Complete**: The test run is completed. This section also displays the time taken for test run.
   - During the **Running** phase, the following step result indicators are displayed with the time taken to process each step:
     - A checkmark appears for the passed step.
     - An error mark appears for the steps with an error.
     - A crossmark appears for the failed step.
     - An info icon appears for the **With** step that indicates a new screen or window.
   - The **Results** tab displays the test as **In Progress** during the test run.
   - After the test run is completed, the **Results** tab displays the test as **Pass** or **Fail**.

   The **Result** page supports selecting individual items from the list. When a row is selected, the corresponding high-level information is displayed in the top banner of the Test Editor.

Figure 3. Running a Web UI test



7. Click to open the **Results** page, where you can view the results, reports, and logs of the completed test run.

Results

You have run a Web UI test.

What to do next

You can view the following information:

- The console output, which includes error messages and logs. See Accessing console output on page 787.
- The results, reports, and logs on the **Results** page after the test completes the run.
- The summary of the test run, which includes passed or failed status, count of passed or failed tests or test steps, and the verification points in the functional report. See Viewing a Functional Report on page 1511.

Related information

Viewing the progress of running test assets on page 1444

Accessing the console output

When you run tests, you can access the console output that provides information such as error messages, logs, and the test verdict. You can then use this output to troubleshoot issues by reviewing details, such as session IDs and error messages.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensure that your project contains a test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.yaml` file to open the test.
   The test steps are displayed on the **Test Editor** page.
5. Click the **Console** tab at the bottom of the **Test Editor** page.

   Alternatively, you can click the expand icon .

   The **Console** output view is displayed.
6. Review the console output to identify the error messages and logs.
7. Perform the following actions to manage the console output:

Action	Description
Search a keyword in the console output	Enter a keyword in the **Search console** field, and the corresponding word is highlighted in the console output.
Select all console content	Right-click the console output area and, then select the **Select All** option to select all content.
Copy the selected content	Right-click the selected content and choose **Copy** to copy the content.
Expand console output view	Click the up arrow to expand the console output view for more details.
Restore console output view	Click the down arrow to return the console output page to its original size.
Close console output view	Click the drop-down icon to close the console output view.

Results

You have accessed the console output content.

What to do next

You can perform any of the following tasks:

- Decide on the actions that you might want to take based on the logs.
- Rerun the test asset.

Viewing step-level results and updating tests

You can view the step-level results of each step after running a Web UI test. You can edit and update the steps in test scripts by replacing smartshots.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a Project Owner or Tester in the project. See Managing access to the server projects on page 1735.
- Ensured that you have run the Web UI test.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.

About this task

Test collects the UI data during the test run and smartshots that are captured during the test run can be used to update the failed step. If an object is not identified during the test run, then the step fails. The test steps are indicated in different colors to specify the status of each test step. For the failed step, you can replace the smartshot with the latest one from the last test run. Also, you can update a step by replacing the smartshot and selecting a target element.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click Tests to view the list of tests recorded for the selected application.
5. Click the `.yaml` file to open the test.
   The test steps are displayed on the **Test Editor** page.

   Note: If you have just run the test and the edit branch is not published, then you can click the color-coded step result indicators that are displayed for each step to view the result details.

6. Click any test step for which you want to view the result details.
   The step details pane is displayed.

7. Click the **Show Result** toggle button in the upper-right corner of the step details pane to view the outcome of the step.

   The **Result** section displays the following details:

   - Step run status, such as **Completed step** or **Failure**, along with the **Duration** is displayed. The failure reason is displayed when the step fails.
   - **Start Time** and **End Time**, which you can view by clicking the **Expand view** icon .
   - Overall step outcome, such as **PASS** or **FAIL**.

   Note: The **Show Result** option can be enabled only if the test has already been run. You can disable this option and navigate to the **Test Editor** page.

8. Click the **Fullscreen** icon to view the details with smartshot and other configuration details in full-screen mode for better visibility.

   Note: The **Fullscreen** icon is displayed only after you click the **Expand view** icon.

9. Click the **Exit fullscreen** icon to close the full-screen view.

   Note: The **Exit fullscreen** icon is displayed only when fullscreen mode is enabled.

10. Click the **Previous Step** icon and the **Next Step** icon to navigate between the previous and next test steps.
11. Perform the following actions to update the step by replacing the smartshot for the step:
    You can update both passed and failed steps.

    a. Click the **Replace Smartshot** icon that is displayed if the edit branch is active.

       The **Replace Smartshot** pop-up window is displayed with a list of smartshots under two different sections: **Result Smartshots** and **Application Smartshots**. The **Result Smartshots** displays the smartshots from the last test run, and the **Application Smartshots** displays all the available smartshots captured earlier for the application. You can hover over an image and click the zoom icon or to view the enlarged image. The image is displayed with the controls to rotate right, rotate left, zoom in, and zoom out.

Note: If the test is run as part of the test suite, then the **Replace Smartshot** icon is disabled even when the edit branch is active. You can click the Open button that is displayed to open the test for editing.

b. Select a smartshot, and then click **Replace** if you want to replace the smartshot for the step.

12. Click the **Save** icon to save the modifications.

Results

You have viewed the step-level result and updated the test.

What to do next

You can perform the following tasks:

- Run the test and view the results. See Running a Web UI test on page 785.
- View the Functional Report to verify the test steps with the variable assignment. See Viewing a Functional Report on page 1511.

Related information

Modification of Web UI tests on page 792

Stopping a test run

You can stop a test run when issues occur or if you decide to stop the run before completion.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Initiated a test run from the **Test Editor** page.

About this task

You can stop a running test from the **Test Editor** page if the test run is in the **In transition** or **Running** state.

1. Go to the **Test Editor** page.
2. Click the **Stop playback** icon .
   The status of the test is displayed as **Inconclusive** on the **Results** tab and **Stopped by User** on the progress bar.

Results

You have stopped a test run.

What to do next

You can re-initiate the test run from the **Test Editor** page.

Modification of Web UI tests

With **Test Editor**, you can view and customize a Web UI test that is created in Test.

The **Test Editor** page displays the test scripts. The test script lists user actions and UI elements recorded as chronological sequence of events during the recording session. Actions are presented in the natural language and you can modify the test manually.

The **Test Editor** also displays the following sections:

- **Catalog** tab: Displays the catalog elements that can be dragged and dropped to an existing script as steps. The catalog elements are categorized as **General**, **UI**, and **Flow**. For more information on the usage of catalog elements, see New test steps from the General catalog on page 798, New steps for UI-related actions and user actions in a test from the UI catalog on page 806, and Adding conditional logic and redefining the flow in tests from the Flow catalog on page 814.

  Note: If you want to drag and drop a catalog element to a test when you use Chrome or Edge on a Ubuntu virtual machine, you must directly connect to the machine instead of connecting through a desktop sharing system such as VNC.

- **Results** tab: Displays the status of the test run when it is in progress, and the verdict of the previous test runs with a link to the **Results** page to view reports.
- **Variables** pane: Displays the defined variables and provides options to add or edit variables.

When you click a test step, the step details pane that is expandable is displayed with different sections as follows:

- **Smartshot**: Displays screen captures with UI objects from the web application under test. You can select a UI object, and add or modify an action related to the object. This section provides **Zoom in**, **Zoom out**, **Pan**, **Fit to available space**, and **Replace Smartshot** controls to work with images.

  The **Zoom in** icon increases the magnification of the smartshot image by 10%. The **Zoom out** icon decreases the magnification of the smartshot image by 10%. By using the **Focus on object** icon, you can move the smartshot within the available viewing area by dragging the image. While the **Focus on object** mode is enabled, smartshot interactions such as drag-and-drop and context menu actions are disabled.

  The **Full screen**

- **Details**: Displays the object type and the associated value or action depending on the step type. This section also displays values that identify the object. You can use any of the following conditions while defining the target to identify the object: Includes, Ends with, Matches regex, Starts with, Equals, Doesn't equal, Doesn't match regex, Doesn't include, Doesn't start with, Doesn't end with.

  For the XPath that identifies the object, you can also provide a relative XPath when you select **xpath** in the drop-down list of properties in the **Object Identified** section. The element for which you provided an XPath is highlighted in the **Smartshot** view.
- **Advanced**: Displays options to define a timeout for the step and disable the step.
- **Elements**: Displays the elements available on the web page that are captured in the smartshot.
- **Properties**: Displays the properties such as aria label, class, content, exists, href, id, label, name, placeholder, tagName, type, value, and XPath of the object. The displayed properties are the properties that are captured while recording and they are required to identify the objects during the playback.

The step details pane displays information about the currently selected step and provides options to modify the step. See Editing a Web UI test on page 793.

Editing a Web UI test

You can edit a Web UI test to add new steps, remove the existing steps, or update the step details.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the application for which you recorded a test in the test navigator panel.
   The assets created for the application are displayed.
5. Click Tests to view the list of tests recorded for the selected application.
6. Click the `.yaml` file to open the test.
   The test steps are displayed on the **Test Editor** page.

Note:

The top-level steps and first-level child steps are automatically expanded to provide better visibility.

7. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 9 on page 794.
8. Select an existing edit branch or create a new edit branch.
9. Select a step to view the details.
   The step details pane is displayed. If the step has a user action, then the **Smartshot**, **UI Object** type, **Action**, and **Object Identified By** details are displayed.
10. Perform any of the following actions to modify the step details as required:

Action	Description
Edit the object and action for the step	Perform the following steps: a. Hover over the UI elements in the **Smartshot** view, right-click the required UI element, and then select **Use this element as a step target** to change the object assigned for the step. The values in the **Object Identified By** field are updated based on the selected UI object. You can also select a different UI object from the list if required. b. You can use the available filter options in the **Smartshot** view to filter the displayed UI objects: i. **Filter by HTML**: Displays the identified UI objects based on HTML elements. ii. **Filter by Window**: Displays the identified UI objects based on the application window. iii. **Filter by jQuery UI**: Displays supported jQuery UI controls identified during recording. Supported controls include Button, Radio Button, Checkbox, Menu, Menu Item, Date Picker, Tabs, and Accordion.

Action	Description
	c. Modify the value from the **UI Object** drop-down list if required. d. Select a user action from the **Action** drop-down list. e. **Optional**: Click the **Click here to add relative to** link, specify the relative location in the **Location** field, and specify the relative object in the **UI Object** field. f. **Optional**: Click the **Click here to add alternative identifier** link, and select a target object from the list in the **Target 2** field. Alternatively, you can drag and drop an element from the **Smartshot** view.
Open the test that is embedded in the **Run** step	Right-click the **Run** step and click **Open**. The embedded test is displayed in **Test Editor**.
Select a step or a set of steps	Right-click a step and click **Select**. Checkboxes are displayed for each step to select the required steps. When you right-click a selected step, the **Extract Steps** and **Deselect All** options are displayed.
Disable the step	Click the **Advanced** tab and select the **Disable** checkbox. Alternatively, you can right-click the step and click **Disable**. The option changes to **Enable** for the disabled step.
Copy the step	Right-click the step and click **Copy**.
Paste the step	Right-click a step, and then click **Paste**. The copied step is pasted after the selected step.
Delete the step	Right-click the step, and then click **Delete**.

11. Click to close the step details pane.
12. Click the **Save** icon to save the modifications.

Results

You have edited a Web UI test.

What to do next

You can perform the following tasks:

- Run the test and view the results. See Running a Web UI test on page 785.
- Add additional steps from the catalog. See New test steps from the General catalog on page 798, New steps for UI-related actions and user actions in a test from the UI catalog on page 806, and Adding conditional logic and redefining the flow in tests from the Flow catalog on page 814.
- Create and use variables in the test. See Creating and assigning test variables on page 847.

Adding user actions in a test from smartshots

When you want to add user actions as steps in an existing Web UI test, you can add such steps from smartshots associated with the test. You can select a graphical element in the smartshot and add a user action for that element.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

About this task

The **Test Editor** page displays a hierarchical view of the steps in a test. The step details pane displays the object properties with the associated smartshot. You can add a step as a node and add further actions as sub-steps by selecting any of the widgets in the **Smartshot** view.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 796.
2. Select an existing branch or create a new branch.
3. Drag and drop one of the following element from the **UI** catalog in the **Catalog** tab after the required step.

- Click
- Select
- Type
- Verify
- Assign

A test step is added.

4. Click the newly added step.
   The step details pane is displayed.
5. Select one of the panes and perform the following actions:

Name of the pane	Action
Smartshot	Hover over a UI element in the **Smartshot** view, right-click, and then select one of the following options: ◦ **Use this element as a step target** ◦ **Create user action for this element**: **Click, Select, Type, Drag & Drop, Verify, Assign** Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object Identified By** section.
Elements	Perform the following actions: a. Expand the list of elements in **Elements**. b. Select an element, right-click, and then select one of the following options: ▪ **Use this element as a step target** ▪ **Create user action for this element**: **Click, Select, Type, Drag & Drop, Verify, Assign**
Properties	Perform the following actions: a. Click any of the UI elements in the **Smartshot** view. b. Select a property from the **Properties** list, right-click, and then select one of the following options: ▪ **Add property to step target** ▪ **Create user action with this property**: **Click, Select, Type, Drag & Drop, Verify, Assign** If the target object is already defined, then you can use the **Replace property in step target** option to replace the identifier.

If you selected the **Use this element as a step target** option, the action or condition is assigned to the selected element. Else, if you selected the **Create user action for this element** option with one of the listed actions, a step is added just before the selected step.

The step details pane is displayed with the **Smartshot**, **UI Object** type, **Action**, and **Object Identified By** details. The values displayed are dependent on the object selected.

Alternatively, you can select the **Code** checkbox, and then enter a JavaScript code that returns the target object for the selected action.

6. **Optional:** Click the **Click here to add relative to** link, specify the relative location in the **Location** field, and then specify the relative object in the **UI Object** field.
7. **Optional:** Click the **Click here to add alternative identifier** link, and select a target object from the list in the **Target 2** field.
8. Click the **Save** icon to save the modifications.

Results

You have added user actions as steps from smartshots in a Web UI test.

What to do next

You can perform the following tasks:

- Run the test and view the results. See Running a Web UI test on page 785.
- Add additional steps from the catalog. See New test steps from the General catalog on page 798, New steps for UI-related actions and user actions in a test from the UI catalog on page 806, and Adding conditional logic and redefining the flow in tests from the Flow catalog on page 814.
- Create and use variables in the test. See Creating and assigning test variables on page 847.

New test steps from the General catalog

When you want to add a log statement, validation, variable assignment, or pause the test for a specific duration, you can add the corresponding steps from the **General** catalog.

You can add the following steps from the **General** catalog in the **Catalog** tab to update the test:

Catalog element	Description	Reference
Assign	You can assign a value to an existing variable or a new variable.	Adding an Assign step on page 799
Log	You can insert a custom log statement in a test as a step. The inserted log statement is printed in the **Console** window and the Functional report.	Adding a Log step on page 802

Catalog element	Description	Reference
	For example, you can add a custom statement to demarcate the test steps related to order generation, user creation, or report generation in an application or you can use a log statement to display the value of a variable.	
Pause	You can add Pause as a step in the recorded test for a specified duration. You can specify the pause duration in milliseconds, seconds, minutes, or hours.	Adding a Wait step on page 802
Verify	You can verify the values assigned to a variable. You can compare the values of two variables.	Adding a Verify step on page 803

Adding an Assign step

You can use the **Assign** step to assign values from datasets to test variables or store values extracted from UI elements in variables during a test run.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.
- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.

About this task

You can use the **Assign** step to assign values to variables during a test run. You can assign values from datasets to support data-driven testing or extract values from UI elements and store them in variables. The assigned values can then be used in subsequent test steps to perform flexible data-driven testing across multiple scenarios

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 3 on page 800.

2. Select an existing branch or create a new branch.
3. Drag the **Assign** statement from the **Catalog > General** pane and drop it under a test step as required.
   The **Assign** step is inserted.

   Note:

   You can also use the **Assign** step within a **while** or **do-while** step, so that the **Assign** step retrieves the values from a dataset to process each row of the dataset iteratively and assign the retrieved values to the test variables.

4. Click the **Assign** step.

   The step details pane opens with the **Details** tab which was the default selection.

5. Select the required mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.

6. Click **Yes**.

   Note:
   - The default mode depends on where the **Assign** step is added. When the **Assign** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.
   - You can switch between the available modes by selecting a different option from the **Mode** list.
   - If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.

7. You can perform the actions that are listed in the following table to configure assign during a test run:

Mode	Action
General	Perform the following steps: a. Select an existing dataset in the **Dataset** field. b. Click **Add** to add a variable and assign the value from dataset. c. Select an existing variable from the **Select a variable to assign** drop down. **Note:** You can also select **Create variable** from the list to create a new variable and assign the value from the dataset. The variable is displayed in the **Variables** section. d. Select the column in the dataset from which you want to assign the values during test run. e. Click the **Close** icon to close the run details pane.
UI	Perform the following steps: a. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**. Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section. The **UI Object** type is identified. b. Enter a property name such as Content, Id, or XPath of the object that must be used for variable assignment in **UI Property**. If you entered the property name correctly, the value of the property is displayed under **Extracted Value**. c. Select an existing variable, or Select **Add new variable**, enter a name for the variable, and then click **Add**.

8. Click the **Save** icon to save the modifications.

Results

You can configured the **Assign** step to assign values to test variables during a test run. Based on the selected mode, the Assign step assigns values from a dataset to variables or stores values extracted from UI elements in variables.

What to do next

You can run the test in **Test Editor** and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785.

Adding a Log step

You can insert a custom **Log** statement in a test as a step. The inserted **Log** statement is printed in the **Console** window and the Functional report. For example, you can add a custom statement to demarcate the test steps related to order generation, user creation, or report generation in an application or you can use a **Log** statement to display the value of a variable.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 802.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Log** element under any test step as required from the **General** catalog in the **Catalog** tab.
4. Click the **Log** step.
   The step details pane is displayed.
5. Enter a custom message in the **Log statement** text box.
6. Click the **Save** icon to save the modifications.

Results

You have added a custom message in the test by using the **Log** step from the **General** catalog.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Wait step

You can add **Wait** as a step in the recorded test for a specified duration. You can specify the wait time in milliseconds, seconds, minutes, or hours. Adding a **Wait** step gives the application the required time to load the UI elements and update the screen.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 803.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Wait** element under any test step as required from the **General** catalog in the **Catalog** tab.
4. Click the **Wait** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Specify the amount of time that the step must pause in **Duration**.
   b. Select the unit of time for the timeout in **Unit**.
   c. Alternatively, you can select the **Code** checkbox to enter the time and unit in the **Duration** text box.
   d. **Optional**: Enter a message to indicate the reason for the pause in the **Note** text box.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Wait** step to pause the test for a specific duration from the **General** catalog.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Verify step

You can use the **Verify** step to validate expected conditions during Web UI test run. The Verify step checks whether the application under test behaves as expected by comparing actual runtime values with expected values defined in the test. If the verification fails, the step is marked as failed in the test run results. After you add the **Verify** step from the **General** section of the catalog, you can choose the type of verification to perform by selecting a mode in the step details pane.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 804.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Verify** element under any test step as required from the **General** catalog in the **Catalog** tab.
4. Click the **Verify** step.
   The step details pane is displayed.
5. Select the required verification mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **Verify** step is added. When the **Verify** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.
6. Click **Yes**.

Note:

- The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.
- You can switch between the available modes by selecting a different option from the **Mode** list.
- If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.

7. You can perform the actions that are listed in the following table to configure verification in a **Verify** step during a test run:

Mode	Action
General: You can use **General** mode to verify variables, expressions, and business logic conditions during a test run. This mode validates runtime data instead of UI elements.	Perform the following steps: a. Type or select a variable name in the first column. b. In the operator field, select an option to indicate the basis of comparison of the two operands. The available operators are as follows: ▪ Equals ▪ Not Equals ▪ Greater Than ▪ Greater Than Equals ▪ Less Than ▪ Less Than Equals ▪ Strict Equals ▪ Strict Not Equals ▪ Includes ▪ Not Includes c. In the third column, type or select the name of the variable to be compared with the value of the selected variable in the first column. Additionally, you can add, duplicate or delete rows for verification. Also, you can switch to the JS code view to create or modify the comparison statement.
UI: You can use **UI** mode to verify UI elements and their properties in the application under test during a test run. This mode ensures that UI components are displayed correctly and match expected values.	Perform the following steps: a. Select an object to be verified by dragging and dropping an element from **Smartshot** view, an element from **Elements** list, or a property-value pair from the **Properties** list.

Mode	Action
	Alternatively, you can select an object to be verified from the list in **UI Object**, and then select an identifier such as Class, Content, Label, XPath, or Id of the object from the list. If you follow this method, you must ensure that the object is highlighted in the **Smartshot** view after you enter the values. b. Click **click here to add a row** to provide values to be verified under **Verification Points**. c. Enter the property name and value to be verified in the newly added row. d. **Optional**: Click the **Click here to add relative to** link, specify the relative location in the **Location** field, and specify the relative object in the **UI Object** field. e. **Optional**: Click the **Click here to add alternative identifier** link, and select a target object from the list in the **Target 2** field. Alternatively, you can drag and drop an element from the **Smartshot** view. f. **Optional**: Select the **Override Timeout** checkbox in the **Advanced** tab and increase the timeout value in **Duration** for the object verification.

8. Click the **Save** icon to save the modifications.

Results

You have added a **Verify** step to validate conditions in a Web UI test during a test run.

What to do next

You can run the test in **Test Editor** and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785.

New steps for UI-related actions and user actions in a test from the UI catalog

When you want to add UI-related actions and user actions to update a test, you can add those steps from the UI catalog. UI-related actions are actions such as opening a new page and adding a new section in the test. User actions are actions such as click, select, type, and key press for user inputs.

You can add the following UI-related actions and user actions from the **UI** catalog in the **Catalog** tab to update the test:

Catalog element	Description	Reference
With	You can add a container for a new page, section, or tab on the web page.	Adding a With step on page 807
Open	You can add a step to open a URL.	Adding an Open step on page 808
Click	You can add a click action as a step.	Adding a Click step on page 809
Select	You can add a select action as a step.	Adding a Select step on page 810
Type	You can add a type action for entering the text in a text box as a step.	Adding a Type step on page 811
Press	You can add a press action for window controls or supported keys as a step. For example, you can add a step to minimize a window or press Ctrl + M.	Adding a Press step on page 812

When you insert a step after a **With** or **Open** step by using the catalog elements such as **Assign**, **Click**, **Press**, **Select**, **Type**, and **Verify** from the **UI** catalog, the step is added without a smartshot. You must drag a smartshot to the pane or click the **click here to add smartshot** link to select a smartshot from the **Add Smartshot** pop-up. You also can hover over an image and click the zoom icon to view the enlarged image. The image displays with the controls to rotate right, rotate left, zoom in, and zoom out.

Adding a With step

You can add a container for a new page, section, or tab on the web page by using the **With** step from the **UI** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 3 on page 808.

2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **With** element under any test step as required from the **UI** catalog in the **Catalog** tab.
4. Click the **With** step.

   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Enter a description of the container in the **Note** text box.
   b. Enter a title to identify the container in the **Context** text box.
6. Click the **Save** icon to save the modifications.

Results

You have added a **With** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding an Open step

You can add a step to open a URL that is required for any test step by using the **Open** step from the **UI** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 3 on page 808.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Open** element under any test step as required from the **UI** catalog in the **Catalog** tab.

4. Click the **Open** step.
   The step details pane is displayed.
5. Perform the following steps in the **Details** tab of the step details pane:
   a. Enter the URL of the page to be opened in the **URL** text box.
   b. Select **Attach** to open the URL in a new tab or window.
   c. Enter a title to identify the container in the **Context** text box.
   d. Enter a description about the container in the **Note** text box.
6. Click the **Advanced** tab.
7. **Optional:** Select the **Disable** checkbox if you want to disable the step.
   The selected step is greyed out.
8. **Optional:** Perform the following steps to set the timeout value:
   a. Select the **Override Timeout** checkbox.
   b. Specify the duration of time that Test must wait for the step to be completed in **Duration**.
   c. Select the unit of time for the timeout in **Unit**.

   Alternatively, you can select the **Code** checkbox and enter the timeout and unit in **Duration**.
9. Close the step details pane.
10. Click the **Save** icon to save the modifications.

Results

You have added an **Open** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Click step

You can add a click action as a step by using the **Click** step from the **UI** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 810.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Click** element under any test step as required from the **UI** catalog in the **Catalog** tab.
4. Click the **Click** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Right-click a UI object in the **Smartshot** pane and select **Use this element as step target**.

      The **UI Object** type, **Click** action, and values for **Object identified by** are displayed. Optionally, you can modify the values that identifies the object.

   b. **Optional**: Select the **Click here to add alternative identifier** link to select one more object that works as an alternative target if the first object was not found.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Click** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Select step

You can add a select action as a step by using the **Select** step from the **UI** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 811.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Select** element under any test step as required from the **UI** catalog in the **Catalog** tab.
4. Click the **Select** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Right-click a UI object in the **Smartshot** pane and select **Use this element as step target**.

      The **UI Object** type, **Value**, and values for **Object identified by** are displayed. Optionally, you can modify the values that identifies the object by using one of the following methods:
      - Enter the values in the **Object identified by** fields.
      - Drag and drop an element from the **Elements** section or a Property-Value pair from the **Properties** section.

   b. **Optional**: Select the **Click here to add alternative identifier** link to select one more object that works as an alternative target if the first object was not found.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Select** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Type step

You can add a type action for entering the text in a text box as a step by using the **Type** step from the **UI** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.

- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 812.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Type** element under any test step as required from the **UI** catalog in the **Catalog** tab.
4. Click the **Type** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Right-click a UI object such as text box that is suitable for entering text in the **Smartshot** pane and select **Use this element as step target**.

      The **UI Object** type, **Value**, and values for **Object identified by** are displayed. Optionally, you can modify the values that identifies the object by using one of the following methods:
      - Enter the values in the **Object identified by** fields.
      - Drag and drop an element from the **Smartshot** view or the **Elements** section, or drag and drop a Property-Value pair from the **Properties** section.

   b. **Optional**: Enter a value to be captured as a user input in the **Value** field.
   c. **Optional**: Select the **Click here to add alternative identifier** link to select one more object that works as an alternative target if the first object was not found.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Type** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Press step

You can add a press action for window controls or supported keys as a step by using the **Press** step from the **UI** catalog. For example, you can add a step to minimize a window or press `Ctrl + M`.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 813.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Press** element under any test step as required from the **UI** catalog in the **Catalog** tab.
4. Click the **Press** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Select an option from the **Press** list.

      The **Press** list displays the Back, Close, Enter, Forward, Keyboard, Maximize, Minimize, Refresh, Reset, Restore, and Tab options. If you selected the keyboard option, then the Modifiers and Key fields are displayed.

   b. Select one or more modifier keys such as Cmd (only for Mac), Ctrl, Shift, Alt in **Modifier**.
   c. Enter a name of the key to be used in combination with the modifier or independently in **Key**. For example, you can use Ctrl + Home or Home.
   d. Select the **Native System Input** checkbox to simulate the input keys or special key actions at the screen level on the active window.

      This option is used to handle non-HTML windows such as file browse dialogs. To interact with file browse dialogs, you must keep your machine unlocked and the browser dialog must be in the foreground.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Press** step from the **UI** catalog to add user actions in a Web UI test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding conditional logic and redefining the flow in tests from the Flow catalog

When you want to add conditional logic and redefine the flow to update a test, you can add such steps from the **Flow** catalog. You can add a test inside the test as a step and abort the test after marking it as failed. Also, you can insert the `If`, `While`, and `Do-While` conditions in the test to make a set of steps run only if a specific condition is met.

You can add conditional logic and redefine the flow by adding the following steps from the **Flow** catalog in the **Catalog** tab to update the test:

Catalog element	Description	Reference
Run	You can embed tests within a test and run them. You can add a test after any step within the test.	Adding a Run step on page 814
Fail	You can abort the test and mark the test as failed. For example, you can add this element as a resulting step after checking a condition by using the `If` statement.	Adding a Fail step on page 815
If	You can insert the `If` logic in a test to make a set of steps run only if a specific condition is met.	Adding an If step on page 816
While	You can insert the `While` logic in a test to make a set of steps run in a loop only if a specific condition is met.	Adding a While step on page 818
Do-While	You can insert the `Do-While` logic in a test to make a set of steps run in a loop only if a specific condition is met after the first iteration.	Adding a Do-While step on page 821

Adding a Run step

When you want to add a test inside the test as a step, you can use the **Run** step from the **Flow** catalog.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.

- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 815.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Run** element under any test step as required from the **Flow** catalog in the **Catalog** tab.
4. Click the **Run** step.
   The step details pane is displayed.
5. Perform the following steps in the step details pane:
   a. Select a test from the list in **Path** or drag and drop the `.yaml` file in the designated field.
   b. **Optional**: Select a dataset from the list in **Dataset** or drag and drop the `.csv` file in the designated field.
6. Click the **Save** icon to save the modifications.
7. **Optional:** Click the expand icon to view the embedded test steps.
   The embedded test steps are displayed in greyed-out mode indicating that you can only view the steps. You can click any step to view the step details. If you want to edit the embedded test, you must right-click the **Run** step and click **Open**. The embedded test is displayed in a new tab in **Test Editor**. When you run the test, the step result indicators are displayed for an embedded test as well.

Results

You have added a **Run** step from the **Flow** catalog to embed a test inside another test.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Fail step

You can abort the test and mark the test as failed. For example, you can add the **Fail** element as a resulting step after checking a condition by using the `If` statement.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 816.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Fail** element under any test step as required from the **Flow** catalog in the **Catalog** tab.
4. Click the **Fail** step.
   The step details pane is displayed.
5. Enter a custom message in the **Fail statement** text box.
6. Click the **Save** icon to save the modifications.

Results

You have added a **Fail** step from the **Flow** catalog to mark the test as failed.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding an If step

You can insert the `If` logic in a test to make a set of steps run only if a specific condition is met.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 3 on page 817.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **If** element under any test step as required from the **Flow** catalog in the **Catalog** tab.
4. Click the **If** step.

   The step details pane is displayed.
5. Select the required mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.
6. Click **Yes**.

   Note:
   - The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.
   - You can switch between the available modes by selecting a different option from the **Mode** list.
   - If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.
7. You can perform the actions that are listed in the following table to configure an **If** step during a test run:

Mode	Action
General	Perform the following steps: a. Select a variable in the **Value 1** list. b. In the **Operator** field, indicate the basis of comparison of the two operands. Note that the two operands are strings. The available operators are as follows: ▪ Equals ▪ Greater Than ▪ Greater Than Equals

Mode	Action
	▪ Equals variable ▪ Less Than ▪ Less Than Equals ▪ Strict Equals c. Enter a value to be compared with the value of the selected variable in the **Value 2** list.
UI	Perform the following steps: a. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**. Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section. The **UI Object** type is identified. b. Enter a property name such as Content, Id, or XPath of the object in the **UI Property** field. If you entered the property name correctly, the value of the property is displayed under **Extracted Value**. You can select the **Code** checkbox to enter the If statement in the text box. This allows you to write the condition in the text box in the JS mode.

8. Click the **Save** icon to save the modifications.

Results

You have configured the **If** step to evaluate a condition during a test run. The steps within the If block are run only when the specified condition is met.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a While step

You can insert the `While` logic in a test to make a set of steps run in a loop only if a specific condition is met after the first iteration. The test results display iteration details such as the iteration number, status, and test run time. The test tree displays information from the latest iteration, while the results panel displays either the last successful iteration or the first failed iteration, depending on the test run outcome.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.

- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 819.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **While** element under any test step as required from the **Flow** catalog in the **Catalog** tab.
4. Click the **While** step.
   The step details pane is displayed.
5. Select the required mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **While** step is added. When the **While** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.
6. Click **Yes**.

   Note:
   - The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.
   - You can switch between the available modes by selecting a different option from the **Mode** list.
   - If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.

7. You can perform the actions that are listed in the following table to configure a **While** step during a test run:

Mode	Action
General	Perform the following steps:

Mode	Action
	a. Select a variable in the **Value 1** list. b. In the **Operator** field, indicate the basis of comparison of the two operands. Note that the two operands are strings. The available operators are as follows: ▪ Equals ▪ Greater Than ▪ Greater Than Equals ▪ Equals variable ▪ Less Than ▪ Less Than Equals ▪ Strict Equals c. Enter a value to be compared with the value of the selected variable in the **Value 2** list.
UI	Perform the following steps: a. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**. Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section. The **UI Object** type is identified. b. Enter a property name such as Content, Id, or XPath of the object in the **UI Property** field. If you entered the property name correctly, the value of the property is displayed under **Extracted Value**. You can select the **Code** checkbox to enter the If statement in the text box. This allows you to write the condition in the text box in the JS mode.

8. Click the **Save** icon to save the modifications.

Results

You have added a **While** step that checks whether a specific condition is met in the test from the **Flow** catalog.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a Do-While step

You can insert the `Do-While` logic in a test to make a set of steps run in a loop only if a specific condition is met after the first iteration. The test results display iteration details such as the iteration number, status, and test run time. The test tree displays information from the latest iteration, while the results panel displays either the last successful iteration or the first failed iteration, depending on the test run outcome.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

About this task

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 821.
2. Select an existing edit branch or create a new edit branch.
3. Drag and drop the **Do While** element under any test step as required from the **Flow** catalog in the **Catalog** tab.
4. Click the **Do While** step.
   The step details pane is displayed.
5. Select the required mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **Do While** step is added. When the **Do While** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.
6. Click **Yes**.

Note:

- You can switch between the available modes by selecting a different option from the **Mode** list.
- If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.

7. You can perform the actions that are listed in the following table to configure a **Do While** step during a test run:

Mode	Action
General	Perform the following steps: a. Select a variable in the **Value 1** list. b. In the **Operator** field, indicate the basis of comparison of the two operands. Note that the two operands are strings. The available operators are as follows: ▪ Equals ▪ Greater Than ▪ Greater Than Equals ▪ Equals variable ▪ Less Than ▪ Less Than Equals ▪ Strict Equals c. Enter a value to be compared with the value of the selected variable in the **Value 2** list.
UI	Perform the following steps: a. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**. Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section. The **UI Object** type is identified. b. Enter a property name such as Content, Id, or XPath of the object in the **UI Property** field. If you entered the property name correctly, the value of the property is displayed under **Extracted Value**. You can select the **Code** checkbox to enter the do while condition in the text box in the JS mode.

8. Click the **Save** icon to save the modifications.

Results

You have configured the **Do While** step to evaluate a condition during a test run. The steps within the **Do While** block run at least once and continue to run as long as the specified condition evaluates to true.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Adding a drag-and-drop action in a test from a smartshot

When you want to manually add the drag-and-drop action in a Web UI test, you can do so by editing the recorded test and then add the drag-and-drop action by using a smartshot.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 823.
2. Select an existing branch or create a new branch.
3. Click the test step before which you want to add a drag-and-drop step.
   The step details pane is displayed.
4. Select an element that you want to drag to the new location from one of the following options:
   - The **Smartshot** view
   - The **Elements** list
   - A property-value pair from the **Properties** list
5. Right-click the element and select the **Drag & Drop** option.
   The **Details** tab displays the values in the **UI Object** and **Action** fields. Also, the **Object Identified By** section displays the **From** and **To** tabs. The **From** tab displays the identified target element.
6. Switch to the **To** tab in the **Object Identified By** section.

7. Select the destination element on which you want to drop the previously selected element from one of the following options:
    - The **Smartshot** view
    - The **Elements** list
8. Right-click the element and select the **Use this element as step target (To)** option.
   The details of the target element are populated.
9. Save the test.

Results

You manually added the drag-and-drop action in a Web UI test by using a smartshot.

What to do next

You can run the test and view the results. See Running a Web UI test on page 785.

Configuring timeouts for tests and test steps

You can configure timeouts for tests and individual steps to define the duration of the time that the system should wait before marking an action as failed due to timeout.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

About this task

You can configure timeouts at two levels:

- Test level: You can set a global timeout for the entire test by using the **Configure settings and data** step.
- Step level: You can override the global timeout for individual steps directly within the step settings.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 3 on page 825.

2. Select an existing edit branch or create a new edit branch.
3. Perform any of the following actions:

Action	Description
Configure timeout for the test	Click the **Configure settings and data** step in **Test Editor** The timeout details are displayed with **Duration** and **Unit** fields. By default, the timeout is set to 30 seconds. You can select the **Override Timeout** checkbox to modify this value.
Configure timeout for a test step	Select a step and select the **Advanced** tab. The **Timeout Details** section including the **Duration** and **Unit** fields, is displayed.

4. Select the **Override Timeout** checkbox.
5. Specify the duration of time that Test must wait for the step to be completed in **Duration**.
6. Select the unit of time for the timeout in **Unit**.
7. **Optional:** Select the **Code** checkbox to enter the timeout and unit.
8. Close the step details pane.
9. Click the **Save** icon to save the modifications.

Results

You have configured timeout settings for your test or individual test steps.

Copying test steps from one test to another

When a set of steps in a test is required in another test, you can copy them and paste them into an existing test or an empty test. You can select the required steps from a test to copy, although they are not in a sequence. If there are any variables associated with the selected step, they are also copied with the step.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 826.
2. Select an existing edit branch or create a new edit branch.
3. Right-click a step or a container and click **Select**.
   Checkboxes are displayed for each step.
4. Select the checkboxes to select any step or a container with multiple steps.
5. Right-click a selected step and click **Copy**.
6. Perform one of the following steps:
   - Open a test, right-click a step where the copied steps are required, and then select **Paste**.
   - Open an empty test, right-click the **Configure settings and data** step, and then select **Paste**.
7. Click the **Save** icon to save the modifications.

Results

You have copied a set of steps from a Web UI test to another test.

Related information

Running a Web UI test on page 785

Renaming a test asset

You can rename test assets such as tests, suites, smartshots, and applications in your project by using the **Rename** option.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a Project Owner or Tester in the project. See Managing access to the server projects on page 1735.
- Ensured that your project contains test assets.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.

1. Log in to Test Hub.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.

3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Right-click a test asset that you want to rename from the test navigator panel and select **Rename**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 6 on page 827.
5. Select an existing edit branch or create a new edit branch.
6. Enter a unique name in the **Rename Asset** dialog.
7. Click **Rename**.
   A success message is displayed.

Results

You have renamed a test asset in your project.

Deleting a test asset

You can delete an application, test, suite, or smartshot when it is no longer required in your project. If the test asset that is selected for deletion is referenced in another test or a test suite, or if the selected smartshot is referenced in a test, a warning is displayed.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a Project Owner or Tester in the project. See Managing access to the server projects on page 1735.
- Ensured that your project contains test assets.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.

1. Log in to Test Hub.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Select a test or a smarshot that you want to delete from the test navigator panel.
5. Click the

button, displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 7 on page 828.

6. Select an existing branch or create a new branch.
7. Click the **Delete** icon .
   Alternatively, right-click a test, suite, smarshot, or an application in the test navigator panel, and then click **Delete**.
   Result
   The **Delete Asset** dialog is displayed. This dialog provides information if the selected asset is referenced by the other test asset.
8. Click the **Delete** button to delete the test asset.

Results

You have deleted a test asset from your project.

Related information

Test navigator panel on page 761

Test Editor page on page 763

Creating a test module from an existing set of steps

When a set of steps are repeated and required in multiple tests, you can create a test module for such repeatable steps. You can select the required steps that are in sequence from an existing test and extract them to create a new test module. After extraction, the selected steps are replaced by the **Run** step and the new test module is listed in the test navigator panel.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

About this task

When you extract a set of steps, the resulting asset might be one of the following types:

- A test with the application or page URL details in the **Open** step. You can play back this type of test.
- A module without the application or page URL. You must add the required URL by adding the details in the **Configure settings and data** step or **Open** step.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 829.
2. Select an existing edit branch or create a new edit branch.
3. Right-click a step or a container and click **Select**.
   Checkboxes are displayed for each step.
4. Select the checkboxes to select a set of steps or a container with multiple steps in sequence.
5. Right-click a selected step and click **Extract Steps**.
   The **Create Test from Steps** dialog is displayed. The **Extracted Content** text box displays the list of selected steps and the **Application** field displays the application for which you create the test.
6. Enter a name for the test in **Asset Name**.
7. Select an existing folder from the list in **Location** to save the test assets that you are creating.
   Alternatively, click **New Folder** to create a folder to save the test assets.
8. Click **Create**.
   The selected steps are replaced by the **Run** step for the new test module and the newly created test module is listed in the test navigator panel.

   Note: When you save the test under a different application by using the **Save As** option, on clicking the step, the step details pane displays the `Error loading smartshot` message for all test steps. To resolve this error, you can either manually insert a suitable smartshot from the new application by using the **Replace Smartshot** option in the step details pane, or you can play back the test and save the required smartshots directly from the test results.

Results

You have created a test module by extracting a set of steps from the existing test.

Related information

Editing a Web UI test on page 793

Running a Web UI test on page 785

Run steps on page 814

Test suite management

A suite is a collection of the tests and other suites. You can combine the multiple tests and suites into a single workflow and organize them into a suite.

As part of the test suite management, you can create, edit, run, and delete the suites. When you create a suite, a `.dts.yaml` file is created and listed under the **Suites** folder in the test navigator panel.

You can run the suite either sequentially or in parallel. In a sequential run, the tests and suites within the suite are run sequentially, one after the other. In a parallel run, the tests and suites within the suite are run concurrently. Setting the maximum concurrency value limits how many test instances can run at the same time. When a suite is running, you can click the Show in progress icon to monitor execution and the page displays active test suite instances up to your specified concurrency limit. For example, if a suite contains three tests and the concurrency value is set to two, the progress page will display two instances running simultaneously.

When you run a suite, each child test within the suite generates an individual result. You can open the test from the suite by right-clicking the test and selecting **Open**, and the results of the child test are displayed in the **Results** tab along with the details of the suite that includes this test.

You can rerun the failed tests and suites instead of rerunning the suite. When you run the test suite, you can access the console output to troubleshoot the issues.

Creating a suite

You can create a suite within your project in Test . You can add multiple tests and suites into a single suite for a test run.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the suite assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Perform any of the following actions:
   - Click **Create Asset > Create Suite** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Suite**.
   - Right-click the application in the test navigator panel and select **New > Suite**.

The **Edit Branch** dialog is displayed if there is no edit branch active. Otherwise, go to step 6 on page 831.

5. Select your action to either select an existing Edit branch or create an Edit branch:
    - Perform the following steps to select an existing Edit branch:
        a. Select **Use an existing edit branch**.
        b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

        c. Click **Ok**.
    - Perform the following steps to create the Edit branch:
        a. Select **Create new edit branch**.
        b. Enter a name for the Edit branch in the **Branch Name** field.
        c. Click **Save**.
6. Perform the following steps in the **Create Suite** dialog to create a suite:
    a. Enter a name for the suite in the **Asset Name** field.
    b. Select an existing application to save the suite asset you create.
    c. Click **Done**.
7. Perform any of the following actions to add the tests and suites to the existing suite:
    - Drag the tests and suites from the test navigator panel to **Suite Editor**.
    - Click the **Click to add** option to add the tests and suites in **Suite Editor**.
8. (Optional) To add the tests and suites to the existing suite from other application, drag the test or suite from other application on the test navigator panel to **Suite Editor**.
9. Click the **Save** icon to save the suite.

Results

You have created a suite.

What to do next

You can perform the following tasks:

- Run the test suite and view the results. See Running a suite sequentially on page 834 and Running a suite concurrently on page 835.
- Modify the suite. See Modifying a suite on page 831.

Modifying a suite

You can modify the existing suite within your project in Test by adding, removing, or rearranging the tests and suites in a suite.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that your project contains a suite.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the suite assets.
   The **Overview** page of the project is displayed.
3. Navigate to **Author > Test Editor**.
   The test navigator panel is displayed.
4. Select the suite that you want to modify from the test navigator panel.
   Result
   The suite opens in **Suite Editor**.
5. Click the

button, displayed on the **Suite Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the suite in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed if there is no edit branch active. Otherwise, go to step 7 on page 833.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:

a. Select **Create new edit branch**.
b. Enter a name for the Edit branch in the **Branch Name** field.
c. Click **Save**.

7. Perform any of the following actions to modify the suite in **Suite Editor**:

Action	Description
Add the test or suite	Perform any of the following actions to add the test or suite to the suite: ◦ Drag the test or suite from the test navigator panel to **Suite Editor**. ◦ Click the **Click to add** option to add the test or suite in **Suite Editor**. The test or suite are added to the existing suite.
Open the test or suite	Right-click the test or suite, and then click **Open**. The test opens in **Test Editor** and the suite opens in **Suite Editor**. Alternatively, click any test step for which you want to view the result details. The step details pane is displayed. Click **Open** to open the test in the **Test Editor** for further review or editing. **Note:** If a step has not been run, results are not available for that step in the **Results** view, and the **Open** option is disabled.
Arrange the test or suite	Drag and drop the test or suite. The test or suite are arranged in the order of your preference.
Copy the test or suite	Right-click the test or suite you want to copy, and then click **Copy**. The test or suite are copied to the clipboard.
Paste the test or suite	Right-click anywhere in **Suite Editor**, and then click **Paste**. The test or suite are added to the end of the workflow.
Remove the test or suite	Right-click the test or suite you want to remove from the suite, and then click **Delete**. The test or suite are removed from the existing suite.

8. Click the **Save** icon to save the changes.

Results

You have modified the suite in your project.

What to do next

You can run the test suite and view the results. See Running a suite sequentially on page 834 and Running a suite concurrently on page 835.

Running a suite sequentially

When you want tests to run one after the other, you can run a suite sequentially that maintains the correct order and accuracy of the test results.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensured that a suite is available in your project.
- Added tests or other suites to the suite.

About this task

During the playback, you can run all tests or only the failed tests. The **Continue on Fail** option ensures that the suite continues to run even if a test fails.

As the test runs, the three color-coded boxes display the counts of passed steps, failed steps, and errors. You can view the step details of the first failed step or error step by clicking the respective color-coded boxes. Also, you can view checkmarks for the steps that pass and crossmarks for the steps that fail.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.dts.yaml` file in the list of Suites to open the suite.
   The tests or suites are displayed on the **Suite Editor** page.
5. Click the **Configure settings and data** step.
   The **Settings** pane is displayed.
6. Ensure that the **Run in Parallel** option is unchecked to run the test in sequence.

   Note: If **Run in Parallel** is selected, tests will execute simultaneously rather than sequentially.

7. Verify that the **Continue on Fail** option is selected as the default setting, under the **Options** tab.

Note: If you clear the **Continue on Fail** checkbox, the suite stops when the first test fails.

8. Click the icon, and then select the **Run all > Run all on Default Cluster** option.

   You can also select the **Run all on Personal Agent** option if the personal agent is in a ready state.

   You can view the following stages on the progress bar during the test run:
   - **In transition**: The test setup is in progress.
   - **Running**: The test is actively running.
   - **Complete**: The test run is completed. This section also displays the time taken for test run.
   - During the **Running** phase, you can view the following details:
     - A checkmark after each test or suite that passes, and a crossmark for any failed test or suite.
     - The **Results** tab displays the test as **In Progress** during the test run.
   - After the test run is completed, you can view the following results:
     - The **Results** tab displays the test as **Pass** or **Fail**.
9. Click to open the **Results** page, where you can view the results, reports, and logs of the completed test run.

Results

You have run a suite sequentially.

What to do next

You can view the following information:

- The details of the suites and individual test results from the **Run Details** tab.
- The console output, which includes error messages and logs. See Accessing console output on page 787.
- The test results, reports, and logs from the **Results** page after the test completes the run.
- The summary of the test run, which includes passed or failed status, count of passed or failed tests or test steps, and the verification points in the functional report. See Viewing a Functional Report on page 1511.

Related information

Viewing the progress of running test assets on page 1444

Running a suite concurrently

You can run a suite to run multiple tests and other suites in parallel to expedite the testing process.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensured that a suite is available in your project.
- Added tests or other suites to the suite.

About this task

You can run a suite concurrently to run multiple suites and tests simultaneously across several pods. You can set up the maximum number of concurrent pods, which defines how many suites or test assets can run in parallel. In the default set up, suites run their child assets sequentially on a single pod, with only one test asset running at a time.

When you select the **Run in Parallel** option, the **Continue on Fail** checkbox is disabled. Whether a test passes or fails, all test assets run independently and concurrently across the available pods.

Restriction: To run server suite assets on the Safari browser, you must use DevOps Test Runtime agents. When you use Safari, the tests in the suite run sequentially even when you selected the **Run in Parallel** option.

As the test runs, the three color-coded boxes display the counts of passed steps, failed steps, and errors. You can view the step details of the first failed step or error step by clicking the respective color-coded boxes. Also, you can view checkmarks for the steps that pass and crossmarks for the steps that fail.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.dts.yaml` file in the list of Suites to open the suite.
   The tests or suites are displayed on the **Suite Editor** page.
5. Click the **Configure settings and data** step.
   The **Settings** pane is displayed.
6. Select the **Run in parallel** checkbox to enable multiple tests to run at the same time.
   The **Continue on Fail** option gets disabled.
7. Enter the value in the **Maximum Concurrent Pods** field.

   Note:

   The **Maximum Concurrent Pods** setting defines the number of test assets from a suite that can run in parallel and the minimum value must be two. A lower value reduces server resource usage but increases run time, and a higher value speeds up the run time at the cost of higher resource consumption.

8. Click the icon, and then select the **Run all > Run all on Default Cluster** option.

You can also select the **Run all on Personal Agent** option if the personal agent is in a ready state.

You can view the following stages on the progress bar during the test run:

- **In transition**: The test setup is in progress.
- **Running**: The test is actively running.
- **Complete**: The test run is completed. This section also displays the time taken for test run.
- During the **Running** phase, you can view the following details:
  - A checkmark after each test or suite that passes, and a crossmark for any failed test or suite.
  - The **Results** tab displays the test as **In Progress** during the test run.
- After the test run is completed, you can view the following results:
  - The **Results** tab displays the test as **Pass** or **Fail**.

9. Click **Analyze** to open the **Results** page, where you can view the results, reports, and logs of the completed test run, and perform the following steps:
   a. Verify that a **secondary** label appears for the secondary executions.
   b. Click on secondary and main/primary execution.
   c. Verify that the **Test Log** and **Functional Report** options are displayed only for the main/primary execution and does not display for the secondary executions.

Results

You have run a suite concurrently.

What to do next

You can view the following information:

- The details of the suites and individual test results from the **Run Details** tab.
- The console output, which includes error messages and logs. See Accessing console output on page 787.
- The test results, reports, and logs from the **Results** page after the test completes the run.
- The summary of the test run, which includes passed or failed status, count of passed or failed tests or test steps, and the verification points in the functional report. See Viewing a Functional Report on page 1511.

Related information

Viewing the progress of running test assets on page 1444

Management of DevOps Test Runtime agents on page 1086

Running multiple suites or tests by using the wildcard pattern

When you want to enable a single suite to run multiple tests and suites based on their name, you can use a **Run** step to refer to tests and suites by using wildcard patterns. You can also use these patterns to specify the suites that must be excluded.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Test suite management on page 829 and Creating a suite on page 830.
- Ensured that there are multiple suites available in your project.

About this task

Within a **Run** step, wildcard patterns can be used to reference tests and suites. The following two wildcards are supported:

- **Single Asterisk (`*`)**: Matches zero or more characters within a single folder and it does not match the `/` character.
- **Double Asterisk (`**`)**: Matches zero or more characters including the `/` character. It can be used to match across multiple folder.

You can only match files located in the same folder as the test suite, or within its subfolders. You cannot use patterns to match files located in a different folder for a different application.

The following table lists the sample patterns:

Pattern	What it matches
*.dtx.yaml	All tests located in the same folder as the test suite.
a*.dtx.yaml	All tests starting with the letter "a" in the same folder as the suite.
**/*.dtx.yaml	All tests in subfolders under the folder containing the suite.
**/abc/*	All tests and suites located in any subfolder that is named ‘abc’ under the folder containing the suite.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Perform one of the following actions:
   - Create a new suite, open the suite, and then add a test or suite.
   - Open a suite from the list of Suites.

   The contents are displayed on the **Suite Editor** page.
5. Click a **Run** step.
   The step details pane is displayed.
6. Enter a wildcard pattern in **Path** to specify the tests or suites to be included in the test run.
7. Click **+Add Exclusion** if you want to exclude some of the suites.

A new text box is displayed.

8. Perform one of the following actions to specify the suite or test to be excluded from the test run:
   - Enter the file name.
   - Click the button to browse and select the suite, and then click **Add**.
   - Drag a suite or test from the test navigator panel to **Suite Editor**.
   - Enter a wildcard pattern.
9. Click to close the step details pane.
10. **Optional:** Perform the following steps to enable multiple tests to run at the same time:
    a. Click the **Configure settings and data** step, and then select the **Run in parallel** checkbox.
    b. Enter a value ≥ 2 in the **Maximum Concurrent Pods** field.

    Note: Tests that are resolved by a wildcard pattern run in parallel only if the suite contains a single run step. That is, the only step defined in the suite must be the test with the wildcard expression.

11. Click to save the changes.
12. Click the icon, and then select the **Run all > Run all on Default Cluster** option.

    You can also select the **Run all on Personal Agent** option if the personal agent is in a ready state.

    The **Results** tab displays the progress of the test run. After the test run is complete, the tab displays the verdict, and links to view the functional report, test log, and results page.

Results

You have run multiple suites or tests.

What to do next

You can view the following information:

- The test results, reports, and logs from the **Results** page after the test completes the run.
- The summary of the test run, which includes passed or failed status, count of passed or failed tests or test steps, and the verification points in the functional report. See Viewing a Functional Report on page 1511.

Running suites on multiple browsers and devices

You can now configure a test suite to run across multiple web browsers and supported devices in one consolidated run. Instead of managing separate test runs for each browser and device, you can define all of your target browsers and devices within the configuration of the test suite. This simplifies cross-browser validation, saves time, and provides comprehensive report.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensured that a suite is available in your project.
- Added tests or other suites to the suite.

About this task

You can select the following browsers that are supported for cross-browser testing:

- Chrome
- Firefox
- Edge
- Safari

Also, you can select the following device profiles that are supported:

Device profile	Default browser
Phone (Small)	Edge
Phone (Medium)	Chrome
Phone (Large)	Edge
Phone (X-Large)	Chrome
Tablet (Small)	Edge
Tablet (Medium)	Chrome
Tablet (Large)	Edge
Tablet (X-Large)	Chrome

When the suite runs, all the tests in the suite run sequentially for each selected browser or device.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Double-click the `.dts.yaml` file in the list of Suites to open the suite.
   The tests or suites are displayed on the **Suite Editor** page.
5. Click the

button, displayed on the **Suite Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

Alternatively, right-click the suite in the test navigator panel and select **Edit**.

The **Edit branch** dialog is displayed if there is no edit branch active. Otherwise, go to step .

6. Select your action to either select an existing Edit branch or create an Edit branch:
    - Perform the following steps to select an existing Edit branch:
        a. Select **Use an existing edit branch**.
        b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

        c. Click **Ok**.
    - Perform the following steps to create the Edit branch:
        a. Select **Create new edit branch**.
        b. Enter a name for the Edit branch in the **Branch Name** field.
        c. Click **Save**.
7. Click the **Configure settings and data** step 7 on page 841.
   The **Settings** pane is displayed.
8. Select the required browser and device profiles from the **Profile** drop-down list.
9. Click to close the step details pane.
10. Click to save the changes.
11. Click the icon, and then select the **Run all > Run all on Default Cluster** option.

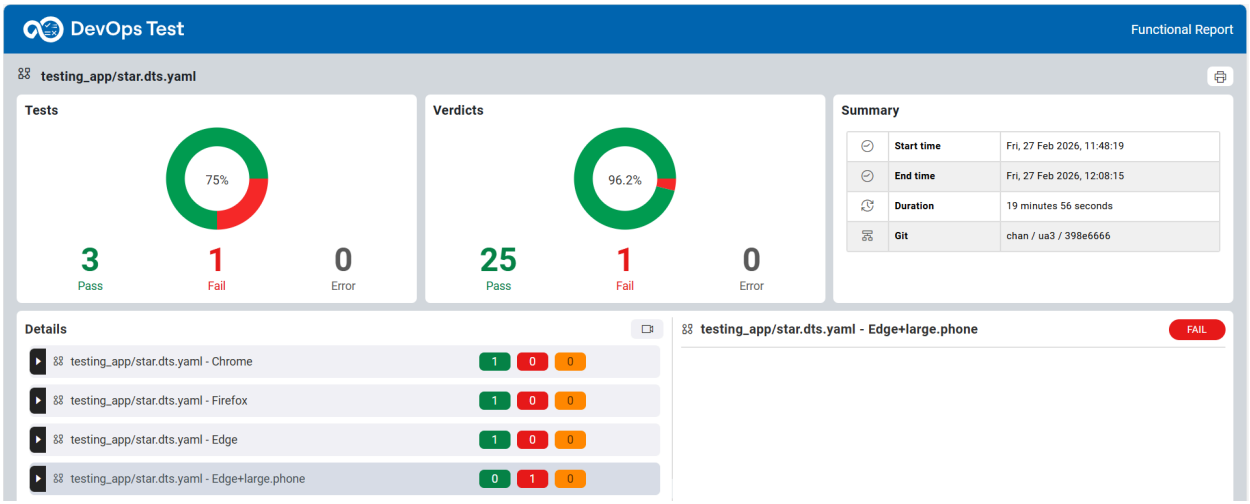
    You can also select the **Run all on Personal Agent** option if the personal agent is in a ready state.

    The **Results** tab displays the progress of the test run. After the test run is complete, the tab displays the verdict and links to view the functional report, test log, and results page.

    Alternatively, you can select the **Run with** option to choose specific browsers. You can select one or multiple browsers. The suite runs only on the selected browser for that run.

12. Click the **Functional Report** icon to open the report in a separate window.

The functional report displays the results of the test run for each configured browser.



Results

You have run the test suite on multiple browsers and devices.

Accessing the console output

When you run suites, you can access the console output that provides information such as error messages, logs, and the test verdict. You can then use this output to troubleshoot issues by reviewing details, such as session IDs and error messages.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensure that your project contains a test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.yaml` file in the list of Suites to open the suite.
   The test steps are displayed on the **Suite Editor** page.
5. Click the **Console** tab at the bottom of the **Test Editor** page.

   Alternatively, you can click the expand icon .

   The **Console** output view is displayed.

6. Review the console output to identify the error messages and logs.
7. Perform the following actions to manage the console output:

Action	Description
Search a keyword in the console output	Enter a keyword in the **Search console** field, and the corresponding word is highlighted in the console output.
Select all console content	Right-click the console output area and, then select the **Select All** option to select all content.
Copy the selected content	Right-click the selected content and choose **Copy** to copy the content.
Expand console output view	Click the up arrow to expand the console output view for more details.
Restore console output view	Click the down arrow to return the console output page to its original size.
Close console output view	Click the drop-down icon to close the console output view.

Results

You have accessed the console output content.

What to do next

You can perform any of the following tasks:

- Decide on the actions that you might want to take based on the logs.
- Rerun the test asset.

Stopping test runs within a suite

You can stop the tests that run within a suite when issues occur or if you decide to stop the run before completion.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Initiated a test run from the **Suite Editor** page.

About this task

You can stop multiple running tests from the **Suite Editor** page if the test run is in **In transition** or **Running** state.

1. Go to the **Suite Editor** page.
2. Click the **Stop playback** icon .
   The status of the tests is displayed as **Inconclusive** on the **Results** tab and **Stopped by User** on the progress bar.

Results

You have stopped test runs within a suite.

What to do next

You can re-initiate the test runs from the **Suite Editor** page.

Rerunning failed tests in a suite

When you run tests in a suite, a few tests might fail. You can rerun the failed tests instead of rerunning the suite.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Ensured that a suite is available in your project.

About this task

The **Run only failed** option is available only if the suite ran previously and contained failed tests. When the **Run only failed** option is available, you can select it to rerun the tests that failed.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the `.dts.yaml` file in the list of Suites to open the suite.
   The tests, suites or both are displayed on the **Suite Editor** page.
5. Click the icon, and then select the **Run failed tests > Default Cluster** option.

   You can also select the **Personal Agent** option if the personal agent is in a ready state.

   You can view the following stages on the progress bar during the test run:

- **In transition**: The test setup is in progress.
- **Running**: The test is actively running.
- **Complete**: The test run is completed.
- During the **Running** phase, you can view the following details:
  - A checkmark after each test or suite that passes, and a crossmark for any failed test or suite.
  - The **Results** tab displays the test as **In Progress** during the test run.
- After the test run is completed, you can view the following results:
  - The **Results** tab displays the test as **Pass** or **Fail**.

6. Click to open the **Results** page, where you can view the results, reports, and logs of the completed test run.

Results

You have rerun the failed tests in a suite.

What to do next

You can view the following information:

- The details of the suites and individual test results from the **Run Details** tab.
- The console output, which includes error messages and logs. See Accessing console output on page 787.
- The test results, reports, and logs from the **Results** page after the test completes the run.

Deleting a suite

You can delete a suite when it is no longer required in your project.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource. See Managing access to the server projects on page 1735.
- Ensured that your project contains a suite.

1. Log in to Test Hub.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the suite assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The test navigator panel is displayed.
4. Select the suite that you want to delete from the test navigator panel.
   Result
   The suite opens in **Suite Editor**.

5. Click the button displayed on the **Suite Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   The **Edit branch** dialog is displayed if there is no edit branch active. Otherwise, go to step 7 on page 846.
6. Select an existing branch or create a new branch.
7. Click the **Delete** icon .

   Alternatively, right-click the suite in the test navigator panel and select **Delete**.

   Result

   The **Delete Asset** dialog is displayed.
8. Click **Delete**.

Results

You have deleted a suite from your project.

Related information

Test navigator panel on page 761

Suite Editor page on page 764

Test variables

A test variable is a user-defined, name-value pair that stores and refers to information throughout a test. You can assign the value of a variable to a test element or assign the value of a UI object property to a test variable.

When you record a test, the URL of the application is stored in the default variable, APP_APP1_URL. You can create variables by using the following options:

- By using the **Add** option in the **Configure settings and data** pane.
- By using the **Add new variable** option when you assign the property of an object to a variable in the **Assign** step.
- By adding var_name: var_value under vars: section in the **Code** view of the test script in **Test Editor**.

You can use the variables for substitution. To substitute data from a test variable, you can use the **Details** section of the step details pane. You can perform the following actions with test variables:

- Provide a default value to the variable during variable creation.
- Change the value of a UI object by replacing the value with a variable. The UI object assumes the value of the variable when the test is run.
- Change the value of the variable using the 'Assign' statement. You can assign an object property to a variable.
- Modify the value of a variable by using the values extracted through RegEx patterns.

Types of variables

Variables are classified as input, local, or output variables based on their purpose and scope.

- **Input variables** : Provide a way to receive values from calling tests or configurations. Use them to pass data into a test or module at runtime.
- **Local variables** : Define values that are used within a test or a suite. These variables remain internal and are not accessible outside the test or suite in which they are declared.
- **Output variables** : These variables are used to return values to the calling test or a suite. You can map these variables to existing or new variables to capture the returned data.

Random number generation

You can generate random numbers and assign the value to a variable by defining a random expression. For example, you can define `${Math.floor(Math.random()*100)}` and assign it to a variable, `${value}`.

You can use a random expression in the following ways:

- Create a new variable in the **Configure settings and data** step, and assign the random expression to the variable.
- Create a new variable in the test, and insert an **Assign** step before any step where the random number is needed. Then, in the **Assign** step details, you can assign the random expression to the variable.

Assigning the random expression to the variable allows the random number to be stored in the variable and referenced throughout the test run.

Creating and assigning test variables

You can create a variable and initialize it to an alphanumeric value. Then, within the test, you can assign the value of the variable to a test element.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 848.
2. Select an existing branch or create a new branch.
3. Click the **Configure settings and data** step in **Test Editor**.
   The **Configure settings and data** pane is displayed.
4. Click **Add** icon

in the **Variables** section.
5. Enter a name for the variable in **Name**.
   The name must start with an alphabet, $, or _ and the name must not be a reserved keyword or contain a space.
6. Enter a value for the variable.
   The newly created variable is listed in the **Variables** section.
   The default variable type is **Input**.
7. **Optional:** Click the variable icon to change the type to **Output** or **Local**.
8. Perform the following steps to assign a variable to a test element:

   a. Select a **Type** or **Select** step for which you want to replace the value with the value of a variable.
      The step details pane is displayed.

   b. Enter the variable name in the **Value** field for the selected **UI Object** in the following format:
      ${variable_name}

      For example, you can replace the text that is saved in the text field for `username` with the value of the variable, `user_name`.

   c. Close the step details pane.
      The test step is updated with the variable that you chose.

9. Click the **Save** icon to save the modifications.

Results

You have created a variable and assigned the value of the variable to a test element.

Note: The variables that you created in **Test Editor** are listed in the **Variables** section of the **Configure settings and data** pane as test variables.

What to do next

- Run the test in **Test Editor** and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785.
- Start the test run from the **Execution** page, and in the **Execute test asset** dialog, navigate to the **Configure settings and data** option on the **Test Editor** page to view the Test variables defined in the test. See Configuring a run of a Web UI test on page 1309 and Variables on page 1317.

Related information

Viewing a Functional Report on page 1511

Assigning an object property to a test variable

You can use the value of a UI object property and assign the value to a test variable in the Web UI tests. The process of assigning a variable to an object property is called a variable assignment.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

About this task

You can modify the value that is being assigned to the variable by using a regular expression. You can extract a substring from the content of an object property by using a regular expression, and then assign the substring to a variable. For example, you can extract *User Documentation* from the Content property of a hyperlink, and then assign the string to the *trimVar* variable. Further, you can include *trimVar* in the subsequent steps and run the test. You can view the variable value *User Documentation* in the Functional Report.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 850.
2. Select an existing branch or create a new branch.

3. Drag the **Assign** statement from the **Catalog > UI** pane and drop it under a test step as required.
   The **Assign** step is inserted.
4. Click the **Assign** step.
   The step details pane is displayed.
5. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**.
   Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section.

   The **UI Object** type is identified.
6. Enter a property name such as Content, Id, or XPath of the object that must be used for variable assignment in **UI Property**.
   If you entered the property name correctly, the value of the property is displayed under **Extracted Value**.
7. Select one of the following options from the **Assign to** list:
   - Select an existing variable.
   - Select **Add new variable**, enter a name for the variable, and then click **Add**.
8. **Optional:** Select the **Modify value** checkbox.
   The **Regex** field is displayed.
9. **Optional:** Enter a regular expression to extract a substring from the value of the object property in **Regex**.
   For example, if you want to extract the last 3 digits from `560064`, which is a Zip code, and assign it to the variable `'zipCode_3'`, you can use the expression `[0-9]{3}`.
10. Close the step details pane.
11. Click the **Save** icon to save the modifications.

Results

You have assigned the value of a UI object property to a test variable in the Web UI test.

Note: The variables that you created in **Test Editor** are listed in the **Variables** section of the **Configure settings and data** pane as test variables.

What to do next

You can perform the following tasks:

- Run the test in **Test Editor** and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785.
- Start the test run from the **Execution** page, and in the **Execute test asset** dialog, navigate to the **Configure settings and data** option on the **Test Editor** page to view the Test variables defined in the test. See Configuring a run of a Web UI test on page 1309 and Variables on page 1317.

Related information

Viewing a Functional Report on page 1511

Adding an Assign step

You can use the **Assign** step to assign values from datasets to test variables or store values extracted from UI elements in variables during a test run.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.
- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.

About this task

You can use the **Assign** step to assign values to variables during a test run. You can assign values from datasets to support data-driven testing or extract values from UI elements and store them in variables. The assigned values can then be used in subsequent test steps to perform flexible data-driven testing across multiple scenarios

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 3 on page 851.
2. Select an existing branch or create a new branch.
3. Drag the **Assign** statement from the **Catalog > General** pane and drop it under a test step as required.
   The **Assign** step is inserted.

   Note:

   You can also use the **Assign** step within a **while** or **do-while** step, so that the **Assign** step retrieves the values from a dataset to process each row of the dataset iteratively and assign the retrieved values to the test variables.

4. Click the **Assign** step.

   The step details pane opens with the **Details** tab which was the default selection.

5. Select the required mode from the **Mode** list:
   - **UI**
   - **General**

   Note: The default mode depends on where the **If** step is added. When the **If** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.

   A confirmation dialog is displayed, indicating that switching modes clears the current step configuration.
6. Click **Yes**.

   Note:
   - The default mode depends on where the **Assign** step is added. When the **Assign** step is added within a **With** step, it opens in **UI** mode. When it is added outside a **With** step, it opens in **General** mode.
   - You can switch between the available modes by selecting a different option from the **Mode** list.
   - If you switch modes after configuring the step, select the required mode and then click **Yes** in the confirmation dialog. The existing configuration is cleared and the step opens in the selected mode.

7. You can perform the actions that are listed in the following table to configure assign during a test run:

Mode	Action
General	Perform the following steps: a. Select an existing dataset in the **Dataset** field. b. Click **Add** to add a variable and assign the value from dataset. c. Select an existing variable from the **Select a variable to assign** drop down. **Note:** You can also select **Create variable** from the list to create a new variable and assign the value from the dataset. The variable is displayed in the **Variables** section. d. Select the column in the dataset from which you want to assign the values during test run. e. Click the **Close** icon to close the run details pane.
UI	Perform the following steps: a. Hover over a UI element in the **Smartshot** view, right-click, and then select **Use this element as a step target**. Alternatively, you can drag and drop a UI element from the **Smartshot** view to the **Object identified by** section. The **UI Object** type is identified.

Mode	Action
	b. Enter a property name such as Content, Id, or XPath of the object that must be used for variable assignment in **UI Property**. If you entered the property name correctly, the value of the property is displayed under **Extracted Value**. c. Select an existing variable, or Select **Add new variable**, enter a name for the variable, and then click **Add**.

8. Click the **Save** icon to save the modifications.

Results

You can configured the **Assign** step to assign values to test variables during a test run. Based on the selected mode, the Assign step assigns values from a dataset to variables or stores values extracted from UI elements in variables.

What to do next

You can run the test in **Test Editor** and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785.

Assigning test variable values from a dataset by using the Run step

You can assign values from a dataset to test variables by using the **Run** step. This supports dynamic data substitution during test execution, allowing subsequent steps to use unique and data-driven values.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.
- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.

About this task

You can create datasets in Test and assign their values to variables in a test or suite. This process helps you to run data-driven tests consistently with parameters and reduces the need to repeat steps. Dataset values are mapped to variables in a test or suite, to perform flexible data-driven testing across multiple scenarios.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test or suite in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 3 on page 854.
2. Select an existing branch or create a new branch.
3. Click the **Run** step to assign the dataset columns to variables.

   The run details pane opens with the **Details** tab which was the default selection. The variables in the test and suite are displayed in the **Inputs** section.
4. Select an existing dataset in the **Dataset** field.
5. Click the drop-down icon in the variable.

   Result

   The variables in the parent test and column names in the selected datasets are listed.
6. Select the column in the dataset from which you want to assign the values during test run.
7. Click Close icon to close the run details pane.
8. Click the **Save** icon to save the modifications.

Results

You have assigned the dataset to test variables in a test or suite by using the **Run** step.

What to do next

You can run the test or suite in **Test Editor** sequentially or concurrently and view the Functional Report to verify the test steps with the variable assignment. See Running a Web UI test on page 785, Running a suite sequentially on page 834, or Running a suite concurrently on page 835.

Configuring test variables in the Run step of a test or suite

You can configure input and output variables in a **Run** step to pass values into and out of a test during test run.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.

- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.

About this task

When configuring input and output variables in a Run step, the following tooltips help indicate the status of each variable:

- **Using default variable**: The variable uses a predefined default value from the called test.
- **Value expected**: No value or mapping has been provided, and the variable does not have a default value. A value must be provided at runtime.
- **Value provided**: A value has been successfully assigned, either through direct input, variable mapping, or hard coding.
- **Using variable**: The variable is mapped to an existing variable in the parent test.
- **Value ignored**: The variable is configured as an output, but its returned value is not used or mapped in the calling test.
- **Value extracted**: The variable is configured as an output, and its value will be captured from the called test

These tooltips guide you in ensuring that all required values are correctly configured before running the test.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** is not displayed, then go to step 3 on page 855.
2. Select an existing branch or create a new branch.
3. Click the **Run** step in the **Test Editor**.

   The run details pane opens with the **Details** tab selected by default. The variables in the test or suite are displayed in the **Inputs** and **Outputs** section.
4. You can perform following tasks to configure variables:

Task	Action
Define a value	Perform following steps: a. Click **Define variable** icon . b. Type the value in the **Value** field. c. Click **Add**. The tooltip is updated as **Value provided**, and the define variable icon is disabled.

Task	Action
Add a new variable in the suite	Perform the following steps: a. Select the drop-down icon under the variable. b. Select **Add new variable**. The **Add Variable** dialog is displayed. c. Type the name of the variable in the **Name** field. d. Type the value in the **Variable** field. e. Click **Add**. A new variable is created and mapped to the selected variable.
Link the new variable to the parent variable	Perform the following steps: a. Click the drop-down icon under the variable. b. Select an existing variable from the parent test. The tooltip is updated as **Value provided**, and the define variable icon is disabled.
Change the value of a variable	Type the new value directly under the variable name. The tooltip is updated as **Value provided**, and the define variable icon is disabled.
Map output variables	Perform the following steps: a. Click the drop-down icon under the variable. b. Select an existing variable from the child test to capture the output value. The tooltip is updated as **Value extracted**, and the define variable icon is disabled. If no mapping is done, the tooltip is displayed as **Value ignored**.

5. Click the **Close** icon to close the run details pane.
6. Click the **Save** icon to save the modifications.

Results

You have configured input and output variables in the **Run** step of a test or suite.

What to do next

You can run the suite in **Test Editor** sequentially or concurrently and view the Functional Report to verify the test steps with the variable assignment. See Running a suite sequentially on page 834 or Running a suite concurrently on page 835.

Creating a dataset by using test variables

When you want to create a dataset by using variables that are configured in a test or a suite, you can do so in Test Editor. In the dataset, each variable gets displayed as a column, and you can update it by adding values to the columns.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Opened the test in **Test Editor**. See Editing a Web UI test on page 793.

1. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** dialog is not displayed, then go to step 3 on page 857.
2. Select an existing branch or create a new branch.
3. Click the **Configure settings and data** step in **Test Editor**.
   The **Configure settings and data** pane is displayed. If the test contains the configured variables, then they are listed under the **Data** section.
4. Click **+ Create Dataset** in the **Data** section.
   The **Create dataset** dialog is displayed.
5. Perform the following steps in the **Create dataset** dialog:
   a. Select an application for which you want to create a dataset from the **Application** drop-down list.
   b. Select an existing folder in **Location** to save the new dataset.
   c. Enter a name for the dataset in **Name**.
6. Click **Next**.

   All the variables available in the test are listed with a checkbox, and only the **Input** variables are selected.
7. Select the variables to be part of the dataset.

8. Click **Create**.
   A success message is displayed. The dataset created is listed under the logical folder `Datasets` within the selected application.

Results

You have created a dataset by using the variables configured in a test.

What to do next

You can view the dataset and add values to it. See Editing a dataset on page 962.

Related information

Adding an Assign step on page 799

Assigning test variable values from a dataset by using the Run step on page 853

Editing a dataset in Test Editor

You can now create and edit a dataset in **Test Editor** either through the catalog in the **Visual** view or through the written code in the **Code** view. However, for encrypted datasets, switching to the **Code** view is disabled to ensure data confidentiality.

Before you begin

1. Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
2. Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
3. Ensured that you have created a dataset or configured a repository that contains the dataset.

About this task

If you are a project **Owner** or **Tester** in Test , you can perform basic tasks in datasets by right-clicking any row, column, or cell in the dataset to organize your data more efficiently. For example, you can cut, copy, paste, insert, or delete rows or columns, show encrypted data, encrypt column data, or rename column names.

After you edit the dataset, you can save the changes made to the dataset, and then you can publish the dataset to the Git repository. If you save and close the edited dataset, the **Changes** page lists the edited dataset, and later you can publish it to the Git repository for other members to use.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.

4. Select the branch that contains your project with the datasets that you want to edit.

   The projects in the selected branch are displayed in the Test Editor navigator panel. The datasets are listed under the folder you created to contain the dataset within a logical folder named `Datasets`.

5. Click the dataset in the Test Editor navigator panel.

   The dataset opens in the right pane.

   The dataset is displayed in the **Visual** view, where you can add, modify, remove, import, or export data by using the toolbar actions or the right-click menu. If needed, you can switch to the **Code** view to edit the dataset in code form.

   Note: For encrypted columns, switching to the **Code** view is not allowed.

6. Click **Edit**.

   Alternatively, right-click the dataset in the Test Editor navigator panel, and then click **Edit**.

   The **Edit Branch** dialog is displayed.

   If the **Edit Branch** dialog is not displayed, then go to step 8 on page 859.

7. Select an existing branch or create a new branch.
8. Perform the following actions to use the top-level editing controls available in the toolbar:

Editing Options	Actions
Find and Replace	To find: a. Click the **Find and Replace** icon. b. Enter the content that you want to search for in the **Find** field. c. Select any of the following options to refine your search content: ▪ Select the **Case sensitive** icon Aa to search the content that matches the exact letter case entered in the **Find** field. ▪ Select the **Match entire cell contents** icon ab to search for cells that contain only the characters that you have entered in the **Find** field. ▪ Select the **Search using regular expression** icon .* to search the pattern that matches strings. For example, to search a cell that contains any number between 0 to 9, do the following: i. Enter *\d* in the **Find** field. ii. Select the **Search using regular expression** checkbox. iii. Click **Find**.

Editing Options	Actions
	d. Click the **Find** icon. If the text is found, the cell containing that text is selected. e. Click the **Find** icon again to find further instances of the search text. To find and replace: a. Click the **Find and Replace** icon. b. Enter the content that you want to search for in the **Find** field. c. Enter the content that you want to replace in the **Replace** field. d. Select any or all of the following options to find and replace the content more effectively: ▪ Select the **Case sensitive** icon to find the content that is the exact letter case of the content entered in the **Find** field. ▪ Select the **Match entire cell contents** icon to find and replace for cells that contain only the characters that you have entered in the **Find** and **Replace** fields. ▪ Select the **Search using regular expression** icon to find and replace the pattern that matches strings. e. Click the **Replace** icon to replace the individual instances. f. Click the **Replace All** icon to replace every instance of the content throughout the dataset.
Undo	a. Click the **Undo** icon. b. From the list, click the change that you want to undo. The **Undo** option reverts the change you made in the dataset.
Redo	a. Click the **Redo** icon. b. From the list, click the change that you want to redo.
Import	You can import variable data into a dataset from the following sources: ◦ When there is a large amount of data stored in a CSV file ◦ When you use the data fabrication feature to generate test data ◦ When you import data by using a database query **Importing data from a CSV file** a. Click the **Import** icon. b. Select **Import a CSV file** from the drop-down list.

Editing Options	Actions
	c. Click **Select file**, and choose the CSV file that contains variable data to import into the dataset. d. Enable the **First row contains headers** if your CSV file contains the header. e. Select one of the following import options to append or overwrite data in the dataset: ▪ Enable **Overwrite existing data** to add the rows and columns from the beginning of the dataset. ▪ Enable **Append existing data** to add rows and columns to the end of the dataset. f. Click **Import**. **Importing data by using data fabrication** a. Click the **Import** icon . b. Select **Fabricate data** from the drop-down list. c. Select a data definition from the drop-down list. d. Enter the number of rows that you want to generate in the **Number of Rows** field. e. Enter a seed value in the **Seed (optional)** field. f. Enable **Fabricate headers** if your schema contains the headers. g. Select one of the following options to append or overwrite data in the dataset: ▪ Enable **Overwrite existing data** to add the rows and columns from the beginning of the dataset. ▪ Enable **Append existing data** to add rows and columns to the end of the dataset. h. Click **Import**. **Importing data by using a database query** a. Click the **Import** icon . b. Select **Import a Database Query** from the drop-down list. c. Select a database query from the drop-down list.

Editing Options	Actions
	d. Enter the number of rows that you want to generate in the **Number of Rows** field. e. Select one of the following options to append or overwrite data in the dataset: ▪ Enable **Overwrite existing data** to add the rows and columns from the beginning of the dataset. ▪ Enable **Append existing data** to add rows and columns to the end of the dataset.
Export	You can export variable data from a dataset to a CSV file to reuse the data in future tests. You must open the dataset from which you want to export the data. Click the **Export** icon to download the dataset as a CSV file.
Configure	You can configure the dataset by clicking the **Menu** icon , and then selecting **Configure**. In the **Configure Dataset** dialog, you can change the row and column settings and configure the string values in the dataset that contain variable data for tests to use when they run. ◦ Select any of the separator values that you used in the CSV file. The available options are **Comma**, **Semicolon**, **Space**, **Tab**, and **Other**. If the CSV file uses any separator characters that are not listed, select **Other**, and then you can specify the character. For example, if the data in the CSV file is separated by a character #, then select the **Other** option and enter # in the field. ◦ Configure the following options to change the row and column settings: ▪ **Column header** - Use an up-down control button to increment or decrement the value of the column header. ▪ **Data start point** - Use an up-down control button to increment or decrement the value of the data starting pointer. ▪ **Current row** - Use an up-down control button to increment or decrement the value of the current row. ◦ Configure the following options to change the string values in the dataset:

Editing Options	Actions
	▪ **Treat as null** - Enter a string value that is to be treated as null when running the test. ▪ **Treat as empty** - Enter a string value that is to be treated as empty when running the test. For example, when you run the test, and the data *123* in the dataset is to be treated as empty, then you must specify *123* in the **Treat as empty** field. ▪ **Treat empty text as null** - Select this field when you want the dataset that contains any blank cells, and the value of those blank cells to be interpreted as null. ◦ Click **Update** to apply the changes.
Discard	Click the **Menu** icon and select **Discard** to discard the changes made to the dataset.

9. Perform the following actions using the editing options available in the column header:

Editing Options	Actions
Rename column	Hover over the column header and click the **Rename column** icon to update the column name.
Delete column	Hover over the column header and click the **Delete column** icon to delete the column.

10. Perform the following actions to use the cell-level editing options available in the right-click context menu.

Option	Actions
Cut	Right-click the cell and click the **Cut** icon to remove the content from the selected cell and paste it into another cell.
Copy	Right-click the cell and click the **Copy** icon to copy the content.
Paste	Right-click the cell and click the **Paste** icon to insert the copied or cut content.
Insert row above	Right-click the row and click **Insert row above** to add a new row above.
Insert row below	Right-click the row and click **Insert row below** to add a new row below.

Option	Actions
Insert column left	Right-click the column and click **Insert column left** to add a column on the left.
Insert column right	Right-click the column and click **Insert column right** to add a column on the right.
Delete row(s)	Right-click the selected row and click **Delete row(s)**.
Delete column(s)	Right-click the selected column and click **Delete column(s)**.
Show encrypted data	Right-click the encrypted column and click the **Show encrypted data** icon to view the encrypted data. Right-click the decrypted column and select **Hide encrypted data** to encrypt the data again. **Important:** The **Show encrypted data** option is available only when you have added the encrypted dataset to the respective classification. For more information about adding an encrypted dataset to a classification, see Adding an encrypted dataset to a classification on page 977.
Encrypt column data	Perform the following steps to encrypt the column data: a. Right-click any cell in a column that you want to encrypt and select the **Encrypt column data** icon . The **Encrypt Column** window is displayed. b. Enter an encryption key in the **Encryption Key** field to encrypt the data in the column. **Remember:** When you have already encrypted other columns in the dataset, you must enter the same encryption key that you used previously. You can use only one encryption key to encrypt columns in a dataset. **Important:** The encryption keys that you use to encrypt data in a dataset are not stored on the server, nor they can be retrieved from the server. Therefore, you must remember to store the encryption keys in a secure location. You must use the same encryption keys to perform the following operations:

Option	Actions
	 ▪ View the encrypted values ▪ Decrypt data ▪ Enable the use of the encrypted dataset during test runs c. Click **Encrypt Column**. Asterisks are displayed instead of actual data for the encrypted column.
Decrypt column data	Perform the following steps to decrypt the column data: a. Right-click the encrypted cells that display the contents with asterisks, and then select the **Decrypt column data** icon . The **Decrypt Column** window is displayed. b. Enter an encryption key in the **Encryption Key** field to decrypt the data in the column. c. Click **Decrypt Column**. Asterisks are replaced with the actual data in the decrypted column. When you run a test that uses a dataset that contains decrypted data, the variable data is substituted for the original data in the recorded test without prompting for the encryption.
Set as current row	During the test run, if you want variable data to be selected from the current row instead of the first row in a dataset, right-click any cell in a row and select the **Set as current row** icon . **When rows are deleted:** If you delete any row between row 1 and the current row, the current row data is taken from the next row. For example, when you set the current row as 6, and then you delete any row between row 1 to row 6, the current row remains at row 6, but the content of row 7 is moved to row 6. **When rows are inserted:** If you insert any new row between row 1 to the current row, the current row data is taken from the previous row. For example, when you set the current row as 6, and then you insert any row between row 1 to row 6, the current row remains at row 6, but the content of row 5 is moved to row 6.

11. Click the **Save** icon to save the changes made to the dataset or the **Save All** icon in the Test Editor navigator panel to save all changes made to different datasets by using the same Edit branch.
12. Click the **Publish** icon to publish the dataset to the Git repository.

Results

You have edited the dataset in the **Test Editor**.

Multiple datasets in tests and test suites

You can configure multiple datasets in tests or test suites to provide additional data for variables during a test run. You can combine data from multiple datasets which supports flexible data-driven testing across different scenarios.

Overview

When working with datasets in a test or test suite, you can use a primary dataset along with additional datasets to provide values for variables.

The primary dataset is used as the main data source. Additional datasets provide supporting values when required. Therefore, you can use data from multiple datasets without duplicating test logic.

In **Test Editor**, you can add additional datasets when configuring tests and test suites.

You can configure multiple datasets in **Test Editor** as follows:

- Select a dataset as the primary dataset.
- Add one or more additional datasets, which are displayed indented under the primary dataset.
- Add additional datasets only after selecting the primary dataset.

Note: You cannot select the same dataset more than once.

Dataset behavior

The following behavior applies when you assign values to variables:

- Columns from both primary and additional datasets are available for mapping to variables.
- If a column name exists in multiple datasets, the value from the first matching dataset is used for mapping to variables. Datasets are considered in the configured order, and the primary dataset is considered first.
- If the same column exists in multiple datasets, the value from the first matching dataset is used.
- Additional datasets are used to provide values that are not available in the primary dataset.

Assigning test variable values from multiple datasets by using the Assign step

You can assign test variable values from multiple datasets to retrieve values from more than one dataset. This capability supports flexible data-driven testing across different scenarios and reduces the need to repeat test steps.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test.
- Ensured that the required variables have been created and are included in your Web UI test.
- Ensured that your project contains a dataset.

1. Open the required test in **Test Editor**.
2. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.
   If the **Edit branch** is not displayed, then go to step 4 on page 867.
3. Select an existing branch or create a new branch.
4. Drag the **Assign** statement from the **Catalog > General** pane and drop it under a test step as required.
   The **Assign** step is inserted.

   Note:

   You can also use the **Assign** step within a **while** or **do-while** step, so that the **Assign** step retrieves the values from a dataset to process each row of the dataset iteratively and assign the retrieved values to the test variables.

5. Click the **Assign** step.

   The step details pane opens with the **Details** tab.
6. Perform the following steps to add multiple datasets:

   a. Select an existing dataset in the **Dataset** field.
      This dataset acts as the primary dataset. The **Add Dataset** option is enabled.

   b. Click **Add Dataset** and select additional datasets.

      You can add multiple datasets as required.

      If you want to remove a dataset, you can click **Remove Dataset** from the dataset list.

7. Click **Add Variable** to add a variable and assign the value from the dataset.

8. Select an existing variable from the **Select a variable to assign** drop down.

   Result

   The variable is displayed in the **Variables** section.
9. Click the drop-down icon for the variable.

   Note: You can also select **Create variable** from the list to create a new variable and assign the value from the dataset.

   Result

   The variables in the test and column names in the selected datasets are listed.
10. Select the column in the dataset from which you want to assign the values during test run.
11. Click the **Close** icon to close the step details pane.
12. Click the **Save** icon to save the modifications.

Results

You have assigned multiple datasets to test variables in a test by using the **Assign** step.

Assigning test variable values from multiple datasets by using the Run step

You can assign or modify test variable values from multiple datasets to retrieve values from more than one dataset. This capability supports flexible data-driven testing across different scenarios and reduces the need to repeat test steps.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.
- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.

1. Open the required test or suite in **Test Editor**.
2. Click the

button displayed on the **Test Editor** page.

   The **Edit** button appears only when no active edit branch is selected.

   Alternatively, right-click the test in the test navigator panel and select **Edit**.

   The **Edit branch** dialog is displayed.

If the **Edit branch** is not displayed, then go to step 4 on page 869.

3. Select an existing branch or create a new branch.
4. Click the **Run** step to assign the dataset columns to variables.

   The step details pane opens with the **Details** tab. The variables in the test and suite are displayed in the **Inputs** section.

5. Perform the following steps to add multiple datasets:

   a. Select an existing dataset in the **Dataset** field. This dataset acts as the primary dataset.
   This dataset acts as the primary dataset. The **Add Dataset** option is enabled.

   b. Click **Add Dataset** and select additional datasets. You can add multiple datasets as required.

   If you want to remove a dataset, you can click **Remove Dataset** from the dataset list.

6. Locate the variable in the **Inputs** variable section for which you want to assign a value.
7. Hover over the variable value field.

   A drop-down icon is displayed next to the field.

8. Click the drop-down icon for the variable.
   Result
   The variables in the test and column names in the selected datasets are listed.
9. Select the column in the dataset from which you want to assign the values during the test run.
10. Click the **Close** icon to close the run details pane.
11. Click the **Save** icon to save the modifications.

Results

You have assigned multiple datasets to test variables in a test or suite by using the **Run** step.

Test environments

You can define custom environments to manage sets of variables such as application URLs and user credentials, and to configure profile overrides. When you run a test, the variables and override rules defined in the environment dynamically replace the values within your test script. You can easily change testing targets without modifying your test script by using environments because the environment-level configurations take precedence over the default values defined in the script.

You can create, edit, copy, move, and delete the environments. When you create an environment, a `.dte.yaml` file is created and listed under the **Environments** folder for the selected application in the test navigator panel.

Environments that you defined for an application are available for selection in the **ENVIRONMENT** tab of the **Execute test asset** dialog when you initiate a test run.

Profile overrides

In addition to managing variables, you can use environments to configure profile overrides. By using profile overrides, you can substitute a default profile, such as a specific browser or emulated device, with a different target profile, such as a cloud-native device or alternative browser, when you run tests.

In the **Execute test asset** dialog, you can either specify an **Environment** that carries override rules or select a browser or device profile in the **Browser** field. If you specify values in both the **Environment** and **Browser** fields, then the value in the **Browser** field takes priority.

Related information

Test navigator panel on page 761

Creating an environment

You can create an environment to define custom variables that override the variables defined in the test when you run a test or suite. You can also configure profile overrides so that cloud-native devices can replace the default browsers or device profiles when you run a test or suite.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can use environments such as Dev, QA, or Staging to manage a set of variables and profiles with different values. When you run a test, the environment variables dynamically replace the matching variable values, and profile overrides replace the browser or device profiles defined in the test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Navigate to **Author > Test Editor**.
   The **Test Editor** page and test navigator panel are displayed.
4. Perform any of the following actions:

- Click the **Action Menu** icon and select **Environment**.
- Right-click the application folder in the test navigator panel and select **New > Environment**.

The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 6 on page 871.

5. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.
6. Perform the following steps to configure the environment details in the **Create Environment** dialog.
   a. Select a target application from the **Application** drop-down list.
   b. Enter a unique name for your environment in the **Name** field.
7. Click **Done**.

   Result

   The new environment is created, and it opens in the **Test Editor** for further configuration.
8. Perform the following steps to add specific variables in the **Variables** section:
   a. Click **+ Create Variable**.
   b. Enter the name of the variable in the **Name** column.
      For example, `App_URL`, `username`, or `password`.
   c. Enter the corresponding value for the variable in the **Value** column.
   d. Repeat these substeps to add all necessary variables for this environment.
9. Perform the following steps to define overrides for the default profiles in the **Profiles** section:
   a. Click **+ Add Override**.
   b. Select the name of the default browser or device profile from the **Profile** drop-down list.
      For example, select `Phone (Small)` or `Chrome`.

c. Select a device from the list of devices for the configured device cloud from the **Override** drop-down list.

d. Repeat these substeps to add all necessary overrides for this environment.

10. Click the **Save** icon to save the details.

Results

You have created an environment.

What to do next

You can perform the following tasks:

- Run a server test asset. See Running server test assets on page 1321 or Running server test assets on an agent on page 1324.
- Run a server test suite asset. See Running server suite assets on page 1329.

Modifying an environment

You can add, update, or remove variables and profile overrides that are part of the configured environments.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that your project contains an environment.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Navigate to **Author > Test Editor**.
   Result
   The **Test Editor** page and test navigator panel are displayed.
4. Double-click the environment that you want to modify in the test navigator panel, and then click **Edit**.
   Alternatively, right-click the environment in the test navigator panel and click **Edit**.
   The **Edit branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 6 on page 873.

5. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.
6. Perform any of the following steps to modify your variables in the **Variables** section of the environment editor:

Action	Steps
Add a new variable	Click **+ Create Variable**, then enter the new variable name and value.
Edit an existing variable	Click inside the **Name** or **Value** field of the specific variable and type your updates.
Delete a variable	Click the **Delete** icon next to the variable row that you want to remove.

7. Perform any of the following steps to modify overrides in the **Profiles** section of the environment editor:

Action	Steps
Add an override	Click **+ Add Override**, select the name of the default browser or device profile from **Profile**, and then select a device in the configured device cloud from **Override**.
Edit an existing override	Select different values from the **Profile** or **Override** field.
Delete an override	Click the **Delete** icon next to the override row that you want to remove.

8. Click the **Save** icon to save the details.

Results

You have modified the existing environment.

Catalog management

You can use the data catalog in Test to browse and manage data generators used in test data creation. The catalog section provides access to built-in generators, allows you to create custom generators tailored to specific needs, and group them using custom categories.

You can access the data catalog for a project from **Author > Catalog** tab in the Test. The catalog includes:

- **Basic Generators**: You can select any generator from the available generators listed in the **Catalog** tab of a data definition. The generators in turn contain different options and fields that generate data by using a built-in database in Test Hub. See Basic generators on page 874.

  Note: You must add at least one generator as a field in the data definition to generate test data.

- **Custom Generators**: Custom generators are modifications of the built-in generators that you modify any or all the fields in a generator to suit your specific requirements of your test data. The modified generator cannot be saved to the built-in generator and hence, you must save them as custom generators by providing a name to identify the custom generator. See Management of custom generators on page 923.
- **Custom Categories:** You can create a custom category in the catalog to contain the custom generators that you create. See Management of custom categories on page 930.

Basic generators

You can use the built-in basic generators in Data definitions with or without modifying the default configurations of the generators.

The Data fabrication feature of Test contains a collection of built-in data generators available in the **Basic** category on the **Catalog** tab of a Data definition.

Basic generators are the built-in generators of the Data fabrication feature of Test that you can use to generate diverse test data. These generators provide commonly used data types and patterns that you can select and configure in a Data definition to generate realistic test data.

The basic generators that you can use in Test are as follows:

- Address
- Boolean
- Dataset
- DateTime
- Enumerator
- Function
- Name
- Number
- Phone
- Regex

- String
- Text
- UUID

Every basic generator has unique parameters with default features. When you click a basic generator that is added as a field in the Data definition, the corresponding unique parameters with their default configuration are displayed in a new dialog. The new dialog is named after the generator that you select.

You can change the default configurations of any basic generator only when it is added as a field in a Data definition. You can use the changed configurations for any basic generator in the Data definition by saving the Data definition. The changed configuration is applicable only in the Data definition that contains it and is not available for use from the **Catalog** tab.

If you want to save the modified generators, you can create them as custom generators and save them in a custom category in the **Catalog** tab.

You can find details about the parameters in each of the basic generators along with their default configuration and the configurations that you can change or modify. The parameters of the basic generators are as follows:

- Parameters of the Address generator on page 875
- Parameters of the Boolean generator on page 880
- Parameters of the Dataset generator on page 883
- Parameters of the DateTime generator on page 885
- Parameters of the Enumerator generator on page 889
- Parameters of the Function generator on page 893
- Parameters of the Name generator on page 899
- Parameters of the Number generator on page 903
- Parameters of the Phone generator on page 907
- Parameters of the Regex generator on page 911
- Parameters of the String generator on page 914
- Parameters of the Text generator on page 919
- Parameters of the UUID generator on page 922

Parameters of the Address generator

After you add the Address generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Address generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Address generator that contains the existing configuration details for that field are displayed in the **Address** dialog.

The following table provides the list of UI elements of the **Address** dialog with their description, default features, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Address**.	The field is not editable.	NA
The **Distribution** field	Determines whether the addresses are randomly generated in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the addresses to follow a random distribution when generated in the test data. • Clear the **Random** option if you want the addresses to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for addresses using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains addresses with the postal codes and `70%` to another configuration that contains the address with no postal codes, then the address column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines whether duplicate or unique addresses are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated addresses to include duplicate entries. This means that some addresses might appear more than once in the generated data.

UI element	Description	Default feature	Action
			• Clear the **Allowed** option if you want to ensure unique addresses with no duplicates. This will prevent any duplicate addresses from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank addresses in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the secondary addresses fields in the generated data will be empty or blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated addresses when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of addresses to be generated. For example, if you enter `50`, the generated data will display the address column with a 50% occurrence rate.
The **First Line** option	Determines whether the generated addresses include the first line of the address.	The **First Line** option is selected.	• Do not perform any action if you want the generated addresses to include the first line of the address. • Clear the **Allowed** option if you want to exclude the first line from the generated addresses.
The **Second Line** option	Determines whether the generated addresses include the second line of the address.	The **Second Line** option is selected.	• Do not perform any action if you want the generated addresses to include the second line of the address. • Clear the **Allowed** option if you want to exclude the second line from the generated addresses.

UI element	Description	Default feature	Action
The **City** option	Determines whether the generated addresses include the city name.	The **City** option is selected.	• Do not perform any action if you want the generated addresses to include the name of the city. • Clear the **Allowed** option if you want to exclude the name of the city from the generated addresses.
The **Zip Code** option	Determines whether the generated addresses include the postal code.	The **Zip Code** option is selected.	• Do not perform any action if you want the generated addresses to include the postal code. • Clear the **Allowed** option if you want to exclude the postal code from the generated addresses.
The **Country** option	Determines whether the generated addresses include the country name.	The **Country** option is selected.	• Do not perform any action if you want the generated addresses to include the name of the country. • Clear the **Allowed** option if you want to exclude the name of the country from the generated addresses.
The **Add new configuration** tab	You can add additional address configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the	Click the tab to create additional address configurations. For example, you can have one configuration for addresses with the postal code, and another configuration for addresses with no postal code. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		Random option is not selected. • The selected options from the previous configuration remain selected, and the unselected options remain unselected for the following elements: ◦ First Line ◦ Second Line ◦ City ◦ Zip Code ◦ Country	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Address** generator is complete, you can use the **Address** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon to save the new generator with a unique name in the catalog. You can reuse the same configuration at a later time in different scenarios without having to set it up again. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Address** generator dialog.

Parameters of the Boolean generator

After you add the Boolean generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Boolean generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Boolean generator that contains the existing configuration details for that field are displayed in the **Boolean** dialog.

The following table provides the list of UI elements of the **Boolean** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Boolean**.	The name is not editable.	NA
The **Distribution** field	Determines if the Boolean values are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the Boolean values, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the Boolean to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the Boolean values using the **Percentage** field. For example, if you set the percentage distribution of `100%` and provide a value as `True`, then all the generated Boolean values in the corresponding column will be true. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Blank perce**	Determines the frequency of blank values in the corresponding column of the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`.

UI element	Description	Default feature	Action
ntage field			For example, if you enter `10`, approximately 10% of the Boolean values in the corresponding column of the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated Boolean values when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of the Boolean values to be generated. For example, if you enter `50`, the generated data will display the Boolean column with a 50% occurrence rate.
The **Format** section	Determines the additional details about the format of the generated Boolean values for the selected configuration.	NA	Provide information about the format of the Boolean values for the selected configuration. For more details about the section, see The Format section on page 882.
The **Add new configuration** tab	You can add additional configurations for boolean values with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the **Random** option is not selected.	Click the tab to create additional configurations for Boolean values. This is useful when you need different sets of Boolean values in your synthetic data. For example, you can have one configuration for `Is married` with a fixed distribution, and another configuration for `Has children` with a random distribution. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		• The **Value** and **Distribution** fields display the same values that were provided in the previous configuration.	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Boolean** generator is complete, you can use the **Boolean** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Boolean** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Dataset** generator should be equal to `100`.

The Format section

You can add additional details about the format of the generated Boolean values for a selected configuration in the **Format** section of the **Boolean** dialog. The following table provides the list of UI elements of the section with their description, default features, and actions that you can perform in the section to customize the configuration:

UI element	Description	Default feature	Action
The **Value** fields	Contains two editable fields.	Displays `Yes` in the first field and `No` in the second field.	You can modify the default values to any other suitable Boolean values.
The **Distribution** fields	Contains two editable fields. The field determines the proportion of the entered values when you do not select the **Random** option.	Editable when **Random** is not selected. Displays 60% and 40% as default values.	If you did not select the **Random** option, enter a number from `0` to `100` in the **Distribution** field for both values. Note: The total percentage of the distribution corresponding to both values must be equal to 100.
The **Random** option	Determines if the Boolean values are randomly generated or not.	The option is selected.	• Do not perform any action if you want the Boolean values, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the Boolean values to follow a fixed distribution and to be non-random.

Parameters of the Dataset generator

After you add the Dataset generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Dataset generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Dataset generator that contains the existing configuration details for that field are displayed in the **Dataset** dialog.

The following table provides the list of UI elements of the **Dataset** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Dataset**.	The name is not editable.	NA
The **Blank**	Determines the frequency of blank datasets in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`.

UI element	Description	Default feature	Action
percentage field			For example, if you enter `10`, approximately 10% of the fields corresponding to datasets in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated rows with datasets.	Editable.	Enter a number from `0` to `100` in the field to set the percentage of rows with datasets to be generated. For example, if you enter `50`, the generated data will display the datasets column with a 50% occurrence rate.
The **Dataset** field	Allows you to browse and select a dataset file from the system.	NA	Click the **Browse** option to select a dataset file from your system. The selected dataset value is displayed in the **Dataset** field.
The **Column** list	Allows you to select the desired column from the dataset for data generation.	NA	Click the list to select the column from the dataset.
The **Add new configuration** tab	You can add additional dataset configurations with specific dataset and column selections, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable. • The **Dataset** and **Column** fields display the	Click the tab to create additional dataset configurations, and then add the required components for the additional dataset configuration. For example, if the previous configuration included a dataset with the Address column, you can add a new dataset in another configuration to include another dataset with the Email column.

UI element	Description	Default feature	Action
		same values that were provided in the previous configuration.	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Dataset** generator is complete, you can use the **Dataset** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Dataset** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Dataset** generator should be equal to `100`.

Parameters of the DateTime generator

After you add the DateTime generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the DateTime generator of a specific field in the Data definition in the Edit branch, the unique parameters of the DateTime generator that contains the existing configuration details for that field are displayed in the **DateTime** dialog.

The following table provides the list of UI elements of the **DateTime** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **DateTime**.	The name is not editable.	NA
The **Distribution** field	Determines if the date and time are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the date and time, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the date and time to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the date and time using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains the dates of weekdays with a fixed time range and `70%` to another configuration that contains the dates of weekends with a different time range, then the enumerator column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique date and time are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated date and time to include duplicate entries. This means that some of the date and time values might appear more than once in the generated data.

UI element	Description	Default feature	Action
			• Clear the **Allowed** option if you want to ensure a unique date and time with no duplicates. This will prevent any duplicate date and time from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank date and time values in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the date and time values in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated date and time values when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of the date and time values to be generated. For example, if you enter `50`, the generated data will display the column corresponding to the date and time values with a 50% occurrence rate.
The **Pattern** list	Allows you to select the date pattern for data generation.	The default pattern is **mm/dd/yyyy**.	Select the required date pattern from the list of available options.

UI element	Description	Default feature	Action
The **Limit dates** section	Allows you to define limitations on the generated dates.	NA	• Either enter the dates in the **From** and **To** fields in the format of dd/mm/yyyy or select the dates from the corresponding calendar icons to reflect in the fields. • Select the days of the week (S, M, T, W, T, F, S) in the **Exclude** section to exclude those days from the **From** and **To** date fields. • Enter multiple dates in the **Specific dates to exclude** field, separated by commas, to exclude specific dates from the generation.
The **Limit times** section	Allows you to define limitations on the generated times.	NA	Enter the times in the **From** and **To** fields in the AM and PM format.
The **Add new configuration** tab	You can add additional date and time configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the **Random** option is not selected. • The following sections display	Click the tab and then add the required components for the additional date and time configuration. For example, you can have one configuration for weekdays with a fixed time range, and another configuration for weekends with a different time range. **Note:** You can click the delete icon to delete any additional configuration that you do not need.

UI element	Description	Default feature	Action
		the same values that were provided in the previous configuration: ◦ Limit dates ◦ Limit times	
The **Save to catalog** option	After the customization of the **DateTime** generator is complete, you can use the **DateTime** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon . You can reuse the same configuration at a later time in different scenarios without having to set it up again from scratch. See Creating custom generators and categories from Data definitions on page 925.
The close icon	NA	NA	Click the icon to exit the **DateTime** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **DateTime** generator should be equal to `100`.

Parameters of the Enumerator generator

After you add the Enumerator generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Enumerator generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Enumerator generator that contains the existing configuration details for that field are displayed in the **Enumerator** dialog.

The following table provides the list of UI elements of the **Enumerator** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Enumerator**.	The name is not editable.	NA
The **Distribution** field	Determines if the enumerator values are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the enumerator values, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the enumerator values to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the enumerator values using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains the names of colors and `70%` to another configuration that contains the names of shapes, then the enumerator column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Blank perc**	Determines the frequency of blank enumerator values in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the enumerator values in the generated data will be blank.

UI element	Description	Default feature	Action
entage field			
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated enumerated values when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of enumerator values to be generated. For example, if you enter `50`, the generated data will display the enumerator column with a 50% occurrence rate.
The **Format** section	Determines the additional details about the format of the generated enumerator values for the selected configuration.	NA	Provide information about the format of the enumerator values for the selected configuration. For more details about the section, see The Format section on page 892.
The **Add new configuration** tab	You can add additional enumerator value configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage**	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for colors with a fixed distribution, and another configuration for shapes with a random distribution. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		field is editable when the **Random** option is not selected. • The **Value** and **Distribution** fields display the same values that were provided in the previous configuration.	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Enumerator** generator is complete, you can use the **Enumerator** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from scratch. See Creating custom generators and categories from Data definitions on page 925.
The close icon	NA	NA	Click the icon to exit the **Enumerator** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Enumerator** generator should be equal to `100`.

The Format section

You can add additional details about the format of the generated enumerator values for a selected configuration in the **Format** section of the **Enumerator** dialog. The following table provides the list of UI elements of the section with their description, default features, and actions that you can perform in the section to customize the configuration:

UI element	Description	Default feature	Action
The **Value** fields	You can add multiple editable fields and provide desired values.	Displays three editable fields with `RED`, `GREEN`, and `BLUE` as the corresponding values.	You can modify the default values to any other suitable enumerator values or add additional values.
The **Distribution** fields	You can add multiple editable fields. The field determines the proportion of the entered values when you do not select the **Random** option.	Editable when **Random** is not selected. Displays 20%, 30%, and 50% as default values.	If you did not select the **Random** option, enter a number from `0` to `100` in the **Distribution** fields for all the values. **Note:** The total percentage of distribution corresponding to both values must be equal to 100.
The **Random** option	Determines if the enumerator values are randomly generated or not.	The option is selected.	• Do not perform any action if you want the enumerator values, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the enumerator values to follow a fixed distribution and to be non-random.

Parameters of the Function generator

After you add the Function generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Function generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Function generator that contains the existing configuration details for that field are displayed in the **Function** dialog.

The following table provides the list of UI elements of the **Function** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Gene**	Displays the name of the selected generator, that is, **Function**.	The name is not editable.	NA

UI element	Description	Default feature	Action
rator field			
The **Distribution** field	Determines if the values are randomly generated or not in the function column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the values, when generated in the function column, to follow a random distribution. • Clear the **Random** option if you want the function values to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the Function values using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains the names in uppercase and `70%` to another configuration that contains the names in lowercase, then the Function column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique function values are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated function values to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique function values with no duplicates. This will prevent any duplicate values from appearing in the function column.

UI element	Description	Default feature	Action
The **Blank percentage** field	Determines the frequency of blank function values in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the function values in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated function values when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of function values to be generated. For example, if you enter `50`, the generated data will display the function column with a 50% occurrence rate.
The **Function** section	Determines the additional details about the function for the selected configuration.	NA	Provide information about the function for the selected configuration. For more details about the section, see The Function section on page 897.
The **Add new configuration** tab	You can add additional function configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage**	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for uppercase names with a 30% distribution and another configuration for lowercase names with a 70% distribution. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		field is editable when the **Random** option is not selected. • The **Function** field displays the same values that were provided in the previous configuration.	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Function** generator is complete, you can use the **Function** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Function** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Function** generator should be equal to `100`.

The Function section

You can add additional details about the function for a selected configuration in the **Function** section of the **Function** dialog. The following table provides the list of UI elements of the section with their description, default features, and actions that you can perform in the section to customize the configuration:

UI element	Description	Default feature	Action
The **Function** field	Combines the built-in methods and the Data definition fields to create custom functions.	Empty	You can either enter a function or select values from the **Fields** and **Methods** scroll areas to create a function.
The **Items for function** field	Finds matching results in both the **Methods** and **Fields** scroll areas based on the entered text.	NA	You can type a few letters in the **Items for function** field, and matching results from both the **Methods** and **Fields** scroll areas will be displayed.
The **Methods** scroll area	Displays predefined Java methods that you can use to perform specific operations on entity fields within the function.	NA	When you double-click on an available method, the corresponding parameter is added to the **Function** field. When you hover over a method in the scroll area, the corresponding function, description, and example are displayed. The Data fabrication feature of Test supports a variety of built-in methods. See The supported built-in methods on page 897.
The **Fields** scroll area	Displays the names of all the fields used in the Data definition. You can double-click a field to add it to the **Function** field and incorporate it into the custom function.	NA	When you double-click an item in the **Fields** scroll area, the corresponding parameter is added to the **Function** field.

The supported built-in methods

The Data fabrication feature of Test supports a variety of built-in methods. The following table provides the list of built-in methods, their description, and relevant examples that you can use as functions:

Method	Description	Example
`replace(oldString, newString)`	Replaces all occurrences of the old string with the new string	`F$address.replace('road', 'street')`
`format(formatString, args)`	Returns a formatted string using format string and arguments	`String.format('Hello %s', F$name)`
`split(matchString)`	Splits the string around the given matching string	`F$name.split(" ")[0].concat('@gmail.com')`. In this example, the name is split where a white space is present and only the first part of the split is considered. Also, the first part of the split is appended with `@gmail.com`. If the input name is `Amy Smith`, then the result of the function is `Amy@gmail.com`.
`join(delimiter, stringElements)`	Joins string elements by a delimiter.	`String.join(',', F$id, F$name)`
`length()`	Returns the length of the string	`F$name.length`
`isBlank()`	Returns True only if the string has zero length or contains only white spaces	`F$address.isBlank`
`trim()`	Removes leading and trailing spaces of the string	`F$phone.trim`
`toUpperCase()`	Converts all characters into uppercase	`F$name.toUpperCase`
`toLowerCase()`	Converts all characters into lowercase	`F$name.toLowerCase`

Method	Description	Example
`concat`	Concatenates the specified string to the end of the selected string	`'Hello '.concat(F$name)`

Parameters of the Name generator

After you add the Name generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Name generator of a specific field in the Data definition in the edit branch, the unique parameters of the Name generator that contains the existing configuration details for that field are displayed in the **Name** dialog.

The following table provides the list of UI elements of the **Name** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Name**.	The name is not editable.	NA
The **Distribution** field	Determines if the names are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the names, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the names to follow a fixed distribution and to be non-random, when you add a new configuration. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.

UI element	Description	Default feature	Action
The **Duplicates** field	Determines if duplicate or unique names are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated names to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique names with no duplicates. This will prevent any duplicate names from appearing in the generated data. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the names using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains the names with titles and `70%` to another configuration that does not contain the title, then the names column of the generated data displays the values accordingly.
The **Blank percentage** field	Determines the frequency of blank names in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the names in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percent**	Determines the proportion of the generated names when	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the

UI element	Description	Default feature	Action
age field	you do not select the **Random** option.		**Percentage** field to set the percentage of names to be generated. For example, if you enter `50`, the generated data will display the names column with a 50% occurrence rate.
The **Title** option	Determines whether the generated names include the title.	The **Title** option is selected.	• Do not perform any action if you want the generated names to include the title. • Clear the **Allowed** option if you want to exclude the title from the generated names.
The **First Name** option	Determines whether the generated names include the first name.	The **First Name** option is selected.	• Do not perform any action if you want the generated names to include the first name. • Clear the **Allowed** option if you want to exclude the first name from the generated names.
The **Last Name** option	Determines whether the generated names include the last name.	The **Last Name** option is selected.	• Do not perform any action if you want the generated names to include the last name. • Clear the **Allowed** option if you want to exclude the last name from the generated names.
The **Add new configuration** tab	You can add additional name configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for names with titles and another configuration for names without titles. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		Random option is not selected. • The selected options from the previous configuration remain selected, and the unselected options remain unselected for the following elements: ◦ Title ◦ First Name ◦ Last Name	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Name** generator is complete, you can use the **Name** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Name** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Name** generator should be equal to `100`.

Parameters of the Number generator

After you add the Number generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Number generator of a specific field in the Data definition in the Edit branch, the unique parameters of the Number generator that contains the existing configuration details for that field are displayed in the **Number** dialog.

The following table provides the list of UI elements of the **Number** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	**Description**	**Default feature**	**Action**
The **Generator** field	Displays the name of the selected generator, that is, **Number**.	The name is not editable.	NA
The **Distribution** field	Determines if the numbers are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the numbers, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the numbers to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the numbers using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains numbers with 2 decimal values and `70%` to another configuration that contains numbers with no decimal values, then the number column of the generated data displays the values accordingly.

UI element	Description	Default feature	Action
			The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique numbers are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated numbers to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique numbers with no duplicates. This will prevent any duplicate numbers from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank numbers in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the numbers in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated numbers when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of numbers to be generated. For example, if you enter `50`, the generated data will display the number column with a 50% occurrence rate.

UI element	Description	Default feature	Action
The **Numbers** section	Determines either the range of numbers or the number of digits that appear in the generated data. You can select either the **Range** or **# of Digits** button and then enter the corresponding values in the **Min** and **Max** fields.	The **# of Digits** button is selected with a minimum digit of **1** and a maximum digit of **10**.	• Enter numeric values in the **Min** and **Max** fields to set the minimum and maximum number of digits respectively. The **# of Digits** button is selected as the default feature. • Click the **Range** button to switch to range-based number generation and then enter numeric values in the **Min** and **Max** fields to set the minimum and maximum range of numbers.
The **Decimal places** section	Determines the range of decimal places that are displayed in the generated numbers.	The **Min** and **Max** fields in the section are blank.	Enter numeric values in the **Min** and **Max** fields to set the minimum and maximum number of decimal places, respectively, in the generated numbers.
The **Add new configuration** tab	You can add additional configurations for numbers with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage**	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for numbers with fixed decimal places and another configuration with no decimal places. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		field is editable when the **Random** option is not selected. • The fields or options in the following sections display the same values that were provided in the previous configuration: ◦ Numbers ◦ Decimal places	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Number** generator is complete, you can use the **Number** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close	NA	NA	Click the icon to exit the **Number** generator dialog.

UI element	Description	Default feature	Action
icon			

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Number** generator should be equal to `100`.

Parameters of the Phone generator

After you add the Phone generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Phone generator of a specific field in the Data definition in the edit branch, the unique parameters of the Phone generator that contains the existing configuration details for that field are displayed in the **Phone** dialog.

The following table provides the list of UI elements of the **Phone** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Phone**.	The name is not editable.	NA
The **Distribution** field	Determines if the phone numbers are randomly generated in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the phone numbers, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the phone numbers to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the phone numbers using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that

UI element	Description	Default feature	Action
			contains a specific pattern with the UK as the country and `70%` to another configuration that contains another pattern with France as the country, then the column of the phone number of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique phone numbers are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated phone numbers to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique phone numbers with no duplicates. This will prevent any duplicate phone numbers from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank phone numbers in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the phone numbers in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percent**	Determines the proportion of the generated phone numbers when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of phone numbers to be generated.

UI element	Description	Default feature	Action
age field			For example, if you enter `50`, the generated data will display the phone numbers column with a 50% occurrence rate.
The **Format** list	Determines the desired format for the generated phone numbers.	NA	Select a specific format from the list.
The **Country** list	Determines the country for which you want to generate phone numbers. <Not sure why this list is provided. Selecting or not selecting from this list has no changes in the generated data - Need to confirm with the team>	NA	Select a specific country name from the list.
The **Add new configuration** tab	You can add additional phone number configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable.	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for colors with a fixed distribution, and another configuration for shapes with a random distribution. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		• The **Percentage** field is editable when the **Random** option is not selected. • The **Format** and **Country** fields display the same values that were provided in the previous configuration.	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Phone** generator is complete, you can use the **Phone** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.

UI element	Description	Default feature	Action
The close icon	NA	NA	Click the icon to exit the **Phone** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Phone** generator should be equal to `100`.

Parameters of the Regex generator

After you add the Regex generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Regex generator of a specific field in the Data definition in the edit branch, the unique parameters of the Regex generator that contains the existing configuration details for that field are displayed in the **Regex** dialog.

The following table provides the list of UI elements of the **Regex** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Regex**.	The name is not editable.	NA
The **Distribution** field	Determines if the Regex values are randomly generated in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the Regex values, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the Regex values to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the Regex values using the **Percentage** field.

UI element	Description	Default feature	Action
			For example, if you set the percentage distribution of `30%` to a configuration that contains Regex values of ten numeric values and `70%` to another configuration that contains Regex values of five alphabets, then the Regex column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique Regex values are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated Regex values to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique Regex values with no duplicates. This will prevent any duplicate Regex values from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank Regex values in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the Regex values in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated Regex values when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of the Regex values to be generated.

UI element	Description	Default feature	Action
			For example, if you enter `50`, the generated data will display the column of the Regex values with a 50% occurrence rate.
The **Regex** field	Determines the custom regular expression pattern to generate data that matches the given pattern.	Displays **[a-z]{4}**.	• Edit the default Regex pattern and enter the desired custom Regex pattern. • Perform the following actions to use the helper, if you are not sure of the required Regex format: ◦ Click the **Helper** option. The **Regex helper** dialog appears. ◦ Enter the sample texts in the fields provided, and then click the **Create regex** option. The Regex format is displayed in the **Regex generated** field. ◦ Click **Save** to close the dialog and display the Regex value in the **Regex** field of the generator dialog.
The **Add new configuration** tab	You can add additional configurations of the Regex values with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the **Random** option is not selected. • The **Regex** fields display	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration that contains ten numerals, and another configuration that contains five alphabets. You can similarly customize the values, distribution, and percentage as required. You can click the delete icon to delete any additional configuration that you do not need.

UI element	Description	Default feature	Action
		the same values that were provided in the previous configuration.	
The **Save to catalog** option	After the customization of the **Regex** generator is complete, you can use the **Regex** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Regex** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Regex** generator should be equal to `100`.

Parameters of the String generator

After you add the String generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the String generator of a specific field in the Data definition in the Edit branch, the unique parameters of the String generator that contains the existing configuration details for that field are displayed in the **String** dialog.

The following table provides the list of UI elements of the **String** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **String**.	The name is not editable.	NA
The **Distribution** field	Determines if the strings are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the strings, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the strings to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the strings using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains strings with special characters and `70%` to another configuration that contains strings with no special characters, then the string column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique strings are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated strings to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique strings with no duplicates. This will prevent any duplicate strings from appearing in the generated data.

UI element	Description	Default feature	Action
The **Blank percentage** field	Determines the frequency of blank strings in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the strings in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated strings when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of strings to be generated. For example, if you enter `50`, the generated data will display the string column with a 50% occurrence rate.
The **Minimum characters** field	Determines the minimum number of digits for the string in the generated data.	Displays `10`.	Edit the default number and enter the custom minimum value, if required.
The **Maximum characters** field	Determines the maximum number of digits for the string in the generated data.	Displays `1,000`.	Edit the default number and enter the custom maximum value, if required.
The **Include special characters** option	Determines whether special characters must be included in the string.	The option is not selected.	Select the option if you want to include special characters in the string.

UI element	Description	Default feature	Action
The **Include digits** option	Determines whether digits must be included in the string.	The option is not selected.	Select the option if you want to include digits in the string.
The **Add new configuration** tab	You can add additional string configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for strings with special characters and another configuration for strings with no special characters. You can similarly customize the values, distribution, and percentage as required.

UI element	Description	Default feature	Action
		Random option is not selected. • The fields or options in the following sections display the same values that were provided in the previous configuration: ◦ Minimum characters ◦ Maximum characters ◦ Include special characters ◦ Include digits	**Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **String** generator is complete, you can use the **String** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.

UI element	Description	Default feature	Action
The close icon	NA	NA	Click the icon to exit the **String** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **String** generator should be equal to `100`.

Parameters of the Text generator

After you add the Text generator in a field of a Data definition, you can customize the existing configurations of the field.

When you click the Text generator of a specific field in the Data definition in the edit branch, the unique parameters of the Text generator that contains the existing configuration details for that field are displayed in the **Text** dialog.

The following table provides the list of UI elements of the **Text** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **Text**.	The name is not editable.	NA
The **Distribution** field	Determines if the texts are randomly generated or not in the corresponding column of the test data. Contains the **Random** option.	The **Random** option is selected.	• Do not perform any action if you want the texts, when generated in the test data, to follow a random distribution. • Clear the **Random** option if you want the texts to follow a fixed distribution and to be non-random. When you clear the option, the **Percentage** fields of all the existing configurations become editable. You can set a fixed distribution for the enumerator values using the **Percentage** field. For example, if you set the percentage distribution of `30%` to a configuration that contains words with a count of 10 and `70%` to another configuration that

UI element	Description	Default feature	Action
			contains words with a count of 5, then the text column of the generated data displays the values accordingly. The percentage for each configuration is automatically rebalanced preserving the allocation in the existing configurations, when you add a configuration subsequently.
The **Duplicates** field	Determines if duplicate or unique texts are present in the generated data. Contains the **Allowed** option.	The **Allowed** option is selected.	• Do not perform any action if you want the generated texts to include duplicate entries. • Clear the **Allowed** option if you want to ensure unique texts with no duplicates. This will prevent any duplicate texts from appearing in the generated data.
The **Blank percentage** field	Determines the frequency of blank texts in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the texts in the generated data will be blank.
The **Name** field	Displays the custom name of the selected configuration.	Displays `Default` and is editable.	Enter an identifiable name to the selected configuration, if required.
The **Percentage** field	Determines the proportion of the generated texts when you do not select the **Random** option.	Editable when **Random** is not selected.	If you did not select the **Random** option for distribution, enter a number from `0` to `100` in the **Percentage** field to set the percentage of texts to be generated. For example, if you enter `50`, the generated data will display the text column with a 50% occurrence rate.
The **Text**	Determines the desired text type for the generated texts.	The **Words** option is selected.	Select a specific text type from the list. The available options are as follows:

UI element	Description	Default feature	Action
type list			• Words • Sentences • Paragraphs
The **Count** field	Determines the count with respect to the selected text type.	Displays `10`.	Edit the default count to enter the custom count of the selected text type.
The **Add new configuration** tab	You can add additional text configurations with specific components and settings, if required.	• The **Name** field displays `Configuration-1` and is editable. • The **Percentage** field is editable when the **Random** option is not selected. • The **Text type** and **Count** lists display the same values that were provided in the previous configuration.	Click the tab and then add the required components for the additional configuration. For example, you can have one configuration for words with a count of 10, and another configuration for words with a count of 5. You can similarly customize the values, distribution, and percentage as required. **Note:** You can click the delete icon to delete any additional configuration that you do not need.
The **Save to catalog** option	After the customization of the **Text** generator is complete, you can use the **Text** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon. You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.

UI element	Description	Default feature	Action
The preview icon	The preview icon is available for each configuration of a generator.	NA	Click the preview icon to view the sample test data for the selected configuration of a generator.
The close icon	NA	NA	Click the icon to exit the **Text** generator dialog.

Note: The total percentage displayed in the **Percentage** fields of all the configurations in the **Text** generator should be equal to `100`.

Parameters of the UUID generator

After you add the UUID (Universal Unique Identifier) generator in a field of a data definition, you can customize the existing configurations of the field.

When you click the UUID generator of a specific field in the Data definition in the edit branch, the unique parameters of the UUID generator that contains the existing configuration details for that field are displayed in the **UUID** dialog.

The following table provides the list of UI elements of the **UUID** dialog with their default parameters, and the actions that you can perform in the dialog to customize the parameters for the selected field:

UI element	Description	Default feature	Action
The **Generator** field	Displays the name of the selected generator, that is, **UUID**.	The name is not editable.	NA
The **Blank percentage** field	Determines the frequency of the unique identifiers in the generated data.	Displays `0` and is editable.	Enter a number from `0` to `100`. For example, if you enter `10`, approximately 10% of the unique identifiers in the generated data will be blank.

UI element	Description	Default feature	Action
The **Save to catalog** option	After the customization of the **UUID** generator is complete, you can use the **UUID** generator as the base generator to create a new custom generator.	NA	Click the **Save to catalog** icon . You can reuse the same configuration at a later time in different scenarios without having to set it up again from the scratch. See Creating custom generators and categories from Data definitions on page 925.
The close icon	NA	NA	Click the icon to exit the **UUID** generator dialog.

Management of custom generators

Test provides flexibility to create, modify, and delete custom generators. You can access and modify custom generators to suit your specific data requirements in the **Catalog** page.

You can use the built-in Basic generators as the base to create custom generators that closely mimic the real-world data patterns and distributions. You can customize the parameters of the Base generators and save them as new custom generators to use them for future uses.

The creation, modification, and deletion of custom generators are restricted to the edit branch only. Also, only those members with either owner or tester roles within the project are authorized to create, modify, or delete these generators. This security measure can help to maintain data integrity and consistency while preventing unauthorized changes in Test. After all the modifications are published, other project members can view the changes and updated assets.

Creating a custom generator

You can create custom generators in addition to the existing generators in the Test catalog. You can customize the existing generators and save them as new custom generators to help you create complex real-world scenarios for synthetic data generation. Additionally, custom generators enable the generation of data for specialized fields or domains that may not be readily available in standard datasets.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Read information about the built-in basic generators. See Basic generators on page 874.
- Read about the parameters of the basic generators. See Basic generators on page 874.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Catalog**.
   The **Catalog** page is displayed.
4. Select the branch where you want to store the data generator from the **Branch** list.
5. Perform one of the following actions to open the **Add new generator** dialog:
   - Perform the following actions to use the base generator:
     a. Click a generator from the **Basic** category that you want to use as the base generator.

        The *Generator* dialog is displayed.
     b. Click the **Customize** icon .
   - Click the icon in the **Catalog** page, and then click **Add new generator**.

   Based on the selected **Branch**, the following dialog is displayed:
   - **Edit Branch**: If you selected a branch other than the Edit branch.
   - **Add new generator**: If you selected the Edit branch.
6. Perform one of the following actions in the **Edit branch** dialog, if you selected a branch other than the Edit branch:

Action	Steps	Result
Select an Edit branch	a. Select an Edit branch from the **Select branch** list if the Edit branch already exists. b. Click **Ok**.	The **Add new generator** dialog is displayed.
Create an Edit branch	a. Select **Create new branch**. b. Enter a name for the Edit branch. c. Click **Save**.	The **Add new generator** dialog is displayed.

7. Enter a unique name for the generator in the **Name** field of the **Add new generator** dialog.
8. Select a built-in generator from the **Base generator** list.
   Test displays only the built-in Basic generators as the base generator.
9. Perform one of the following actions to select the category of the new generator in the **Category** list:
   - To select an existing category, go to step 10 on page 924.
   - To add a new category, go to step 11 on page 925.
10. Perform the following steps to select an existing category:

a. Select an existing category, if any, that is displayed in the **Category** list.

Note: You cannot create a new generator in the **Basic** category. Therefore, the **Basic** category is not displayed in the **Category** list.

b. Click **Create**.

The new generator is added under the existing category of the **Catalog** page.

11. Perform the following steps to add a new category:
    a. Click **Add new category** in the **Category** list, and then click **Next**.

    The **Add new category** dialog displays a new set of fields to capture more details about the new category.

    b. Enter a unique name in the **Category name** field where the new generator will be saved in the catalog of Test.
    c. Select a color from the available range of colors so that you can identify the new category easily.
    d. Select the location to create the new category in the selected folder.

    For example, if you entered `MyFolder/MyCategory` in the **Category name** field, and then selected `Location` as the location to create the category in the folder, then the category is created under the `Location/MyFolder/MyCategory` path.

    e. Click **Create**.

    The new generator is added in the newly created category and is displayed in the **Catalog** page.

Results

You have created a custom generator in addition to the existing generators of the Test catalog.

What to do next

You can perform any of the following tasks:

- Publish the custom generators to the project repository so that other members of the project can use the custom generator. See Publishing all changes in an Edit branch on page 1023.
- Modify the parameters of custom generators. See Modifying custom generators on page 927.
- Delete custom generators. See Deleting custom generators on page 929.

Creating custom generators and categories from Data definitions

You can create custom generators and categories from existing Data definitions in Test . After you add an existing generator in a field of a Data definition, you can customize the generator parameters that are displayed for the field. You can then save the customized parameters in the Data fabrication catalog as a new generator. You can also save the new generator either in an existing category or save in a new category.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.
- Read about the parameters of the basic generators. See Basic generators on page 874.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Open a Data definition in the Edit branch.
5. Click a generator in the **Generator** column of a field where you want to customize the configuration.
6. Perform any modification in the *Generator* dialog to customize the existing configuration of the selected field.
7. Click **Save to catalog** to save the customized parameters in the catalog of the Data fabrication feature as a new generator.
   The **Add new generator** dialog is displayed.
8. Enter a unique name for the new generator in the **Name** field.
   Test uses the existing generator as the base generator. Therefore, the **Base generator** list is not editable.
9. Perform one of the following actions to select the category of the new generator:
   - To select an existing category, go to step 10 on page 926.
   - To add a new category, go to step 11 on page 926.
10. Perform the following steps to select an existing category:
    a. Select an existing category that is displayed in the **Category** list.
    b. Click **Create**.

       The new generator is added in the existing category and is displayed in the **Catalog** panel of the data definition.
11. Perform the following steps to add a new category:
    a. Click **Add new category** in the **Category** list, and then click **Next**.

       The **Add new generator** dialog displays a new set of fields to capture more details about the new category.
    b. Enter a unique name in the **Category name** field where the new generator will be saved in the catalog of Test.
    c. Select a color from the available range of colors so that you can identify the new category easily.
    d. Select the location to create the new category in the selected folder.

       For example, if you entered `MyFolder/MyCategory` in the **Category name** field, and then selected `Location` as the location to create the category in the folder, then the category is created under the `Location/MyFolder/MyCategory` path.

e. Click **Create**.

The new generator is added in the newly created category and is displayed in the **Catalog** panel of the data definition.

Results

You have created a custom generator using an existing generator from a Data definition. Also, you have learned to create a new category and save the custom generator under the category.

What to do next

You can save and publish the Data definition to the Git repository so that other members of the project can use the Data definition. See Publishing all changes in an Edit branch on page 1023.

Modifying custom generators

You can modify the parameters of custom generators from the Catalog page of Test . You can make modifications to precisely align the test data that uses the custom generators with the desired dataset.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a custom generator. See Creating a custom generator on page 923.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Catalog**.
   The **Catalog** page is displayed.
4. Select the branch that contains the custom generator for modification from the **Branch** list.
5. Select the custom generator for modification.

   The selected *Generator* dialog that contains the following tabs is displayed:

Element	Description
The **Configuration** tab	Displays all the existing parameters of the selected custom generator.
The **Details** tab	Provides following details about the selected custom generator and its associated Git details: ◦ Name of the branch that contains the generator ◦ Description for the generator, if any

Element	Description
	◦ Git-related details for the generator as follows: ▪ Asset ID: The ID uniquely identifies the generator. You also have the option to copy the ID for easy reference or use. ▪ Repo: The repository name and path where the generator is stored. ▪ Path: The exact location of the generator within the repository. ▪ Commit: The Git commit message, the name of the member who performed the changes, and the date and time of the commit. ▪ The copy button: You can click the button to copy the complete SHA commit ID for the generator. ▪ The commit ID button: You can click the button to view the changes that were made in the commit directly within Git. ▪ The browse button: You can click the button to view the specific location in the Git history where the commit was performed for the generator.
The **Usage** tab	Displays a list of all the Data definitions along with their location in the project that currently use the selected generator.

6. Perform one of the following actions in the **Generator** dialog:
   ◦ Perform the required modifications to the existing configurations in the **Configuration** tab, if you are in the edit branch.
   ◦ Perform the following actions, if you are in a branch other than the Edit branch:

      a. Click the **Edit** icon .

         The **Edit branch** dialog is displayed.

      b. Perform one of the following actions in the dialog:

Action	Steps	Result
Select an Edit branch	i. Select an Edit branch from the **Select branch** list if the Edit branch already exists. ii. Click **Ok**.	The **Catalog** page is displayed in the Edit branch. You can open the custom generator and perform modifications to the existing generator configurations in the **Configuration** tab of the *Generator* dialog.
Create an Edit branch	i. Select **Create new branch**. ii. Enter a name for the Edit branch. iii. Click **Save**.	The **Catalog** page is displayed in the Edit branch. You can open the custom generator and perform modifications to

Action	Steps	Result
		the existing generator configurations in the **Configuration** tab of the *Generator* dialog.

7. Click the save icon in the *Generator* dialog to save the modified custom generator.

Note: If you want to continue using the modified generator to generate test data without publishing it, you must save the changes. However, if you want to collaborate with other members of your project, you must publish the changes.

Results

You have modified the parameters of a custom generator.

What to do next

You can perform any of the following tasks:

- Publish the custom generators to the project repository so that other members of the project can use the custom generator. See Publishing all changes in an Edit branch on page 1023.
- Delete custom generators. See Deleting custom generators on page 929.

Deleting custom generators

You can delete custom generators to remove them when custom generators are outdated or no longer relevant to the current project. Additionally, when custom generators contain errors or inaccuracies that can lead to misleading or incorrect fabricated data, then you might need to remove them. This ensures that the project remains clutter-free, optimizes the performance of Test , and reduces storage requirements.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a custom generator. See Creating a custom generator on page 923.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Catalog**.
   The **Catalog** page is displayed.

4. Select the branch that contains the custom generator to delete from the **Branch** list.
5. Select the custom generator for deletion.
   The selected *Generator* dialog is displayed.
6. Click the **Delete** icon .

Results

You deleted a custom generator from your project.

What to do next

You can publish the changes performed to the custom generator (that is, deletion of the custom generator) to the Git repository so that other members of the project do not see the deleted custom generator. See Publishing all changes in an Edit branch on page 1023.

Management of custom categories

Test provides flexibility to create and delete custom categories in the **Catalog** page. The **Catalog** page displays both the basic and custom categories of generators in a project.

The basic category consists of all the built-in basic generators to simplify the access when you work with Data definitions. When you create custom generators that are tailored to specific use cases, then you can save them under the custom categories and organize them in a way that aligns with the project requirements. You can locate and use the custom generators when you need.

The creation and deletion of custom categories are restricted to the Edit branch only. Also, only those members with either owner or tester roles within the project are authorized to create or delete these categories. This security measure can help to maintain data integrity and consistency while preventing unauthorized changes in Test. After all the modifications are published, other project members can view the changes and updated assets.

Creating a new custom category

You can create custom categories in addition to the existing Basic category in the Test catalog. You can save the custom generators under various custom categories with unique names.

Before you begin

You must be a member of the project with the **Owner** or **Tester** role.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Catalog**.
   The **Catalog** page is displayed.
4. Select the branch where you want to store the custom category from the **Branch** list.

5. Click the icon in the **Catalog** page, and then click **Add new category**.
   Based on the selected **Branch**, the following dialog is displayed:
   - **Edit Branch**: If you selected a branch other than the Edit branch.
   - **Add new category**: If you selected the Edit branch.
6. Perform one of the following actions in the **Edit branch** dialog, if you selected a branch other than the Edit branch:

Action	Steps	Result
Select an Edit branch	a. Select an Edit branch from the **Select branch** list if the Edit branch already exists. b. Click **Ok**.	The **Add new category** dialog is displayed.
Create an Edit branch	a. Select **Create new branch**. b. Enter a name for the Edit branch. c. Click **Save**.	The **Add new category** dialog is displayed.

7. Perform the following steps in the **Add new category** dialog:
   a. Enter a unique name in the **Category name** field.
   b. Select a color from the available range of colors so that you can identify the new category easily.
   c. Select the location to create the new category in the selected folder.

      For example, if you entered `MyFolder/MyCategory` in the **Category name** field, and then selected `Location` as the location to create the category in the folder, then the category is created under the `Location/MyFolder/MyCategory` path.

   d. Click **Create**.

   The new category is added in the **Catalog** page.

Results

You created a custom category in the **Catalog** page.

What to do next

You can perform the following tasks:

- Add custom generators to the new category. See Creating a custom generator on page 923.
- Publish the changes to the Git repository so that other members of the project can use the new custom category. See Publishing all changes in an Edit branch on page 1023.

Deleting custom categories

You can delete custom categories to remove them when custom categories are outdated or no longer relevant to the current project. Additionally, when custom categories contain inaccuracies that can lead to misleading or incorrect fabricated data,

then you might need to remove them. This ensures that the project remains clutter-free, optimizes the performance of Test , and reduces storage requirements.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a custom category. See Creating a new custom category on page 930.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Catalog**.
   The **Catalog** page is displayed.
4. Select the branch that contains the custom category that you want to delete from the **Branch** list.
5. Click the **Delete** icon beside the category tab to delete the custom category.
   Based on the selected **Branch**, the following dialog is displayed:
   - **Edit Branch**: If you selected a branch other than the Edit branch.
   - **Delete category**: If you selected the Edit branch.
6. Perform one of the following actions in the **Edit branch** dialog, if you selected a branch other than the Edit branch:

Action	Steps	Result
Select an Edit branch	◦ Select an Edit branch from the **Select branch** list if the Edit branch already exists. ◦ Click **Ok**.	The **Delete category** dialog is displayed.
Create an Edit branch	◦ Select **Create new branch**. ◦ Enter a name for the Edit branch. ◦ Click **Save**.	The **Delete category** dialog is displayed.

7. Click **Delete** in the **Delete category** dialog.

Results

You deleted a custom category from your project.

What to do next

You can publish the changes performed to the custom category (that is, deletion of the custom category) to the Git repository so that other members of the project do not see the deleted category. See Publishing all changes in an Edit branch on page 1023.

Data management

Effective data management is at the core of robust application testing. In Test , you can manage test data by using the **Data** tab, which provides a centralized workspace for handling various types of data. Whether you work with fabricated data, retrieve information from databases, or define structured schemas, this interface supports both creating new data and reusing existing assets across your testing requirements.

The **Data** tab contains the following core components:

- **Data Definition**: Data definitions represent the data structures that are composed of one or more generators. Test provides you the interface to create and manage data definitions so that you can design data structures that closely resemble the real-world scenarios. You can configure generators, define rules, set restrictions and distributions if required, and generate test data using a data definition. You can generate realistic and diverse test data for comprehensive testing scenarios. See Management of data definition on page 979 and Catalog management on page 874.
- **Dataset**: Datasets provide tests with variable data during a test run. The test that uses a dataset at the run time replaces a value in the recorded test with variable test data that is stored in the dataset. In Test, you can create a dataset and use its values during the run time to test various scenarios. See Management of datasets on page 954.
- **Database Query**: A database query is a request for data or information from a database. You can run queries on integrated databases to examine existing data structures and retrieve records. Then, use the information to guide the design of schemas or enhance your data definitions. See Management of database query on page 947.
- **Schema**: Schemas define how data objects relate to each other. In Test Hub, schemas provide a visual and configurable way to design relational data environments that closely mirror real-world systems. See Management of schemas on page 999.

Related information

Management of datasets on page 954

Management of data definition on page 979

Management of schemas on page 999

Management of database query on page 947

Generating fabricated test data on page 1012

Prerequisites for working with datasets and data definitions

You can find information about the tasks that you need to perform before you work with datasets and data definitions in Test .

Management of classifications

You can create a classification and provide access only for the required members of a project in Test. You can add the encrypted datasets, restricted database connections, or API secrets that you create in a project to the classification and

ensure that it can be accessed and used only by the members who have access to the classification. See Management of classifications on page 934.

Management of database connections

You can import the data that you have in an external database that can be used to create the test assets such as datasets and Data definitions in Test.

You can establish and manage connections with the following external databases in **Test Hub**:

- Oracle
- PostgreSQL
- IBM DB2

See Management of database connections on page 938.

Management of database queries

You can create a database query to import the data that is present in an external database, after you establish a connection with the external database. You can query the required data from a database when you create a database query to import the useful data present in your existing databases to create the test assets. See Management of database query on page 947.

Management of classifications

You can find information about creating, editing, deleting, and granting access to a classification in Test .

You can find the following information about managing the classifications:

- Creating a classification on page 934
- Editing a classification on page 935
- Deleting a classification on page 935
- Granting access to classification on page 936

Creating a classification

As a project *Project Owner* or *Tester*, you must create a classification to organize encrypted datasets, external database connections, and API secrets so that project members can use them efficiently..

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created a project in Test. See Managing DevOps Test projects on page 1724.
- Configured the repository that contains the test assets in your project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click **Create Classification** and enter a name for the classification.
5. Click **Create**.
   The classification is added to the **Classification** list.

What to do next

You can grant access to other project members to use the encrypted dataset. See Granting access to classification on page 936.

Editing a classification

You might want to change the name of a classification that you created, you can do so by editing the classification.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created a classification. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification that you want to edit from the list under **Classification**.
5. Click the **Edit** icon in line with the classification.
   The **Edit classification** dialog is displayed.
6. Edit the name of the classification, and then click **Save**.

Results

You edited the name for a classification in your project.

Deleting a classification

After you create a classification, you can delete a classification when it is not required in your test environment.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created at least one dataset and encrypted the dataset with an encryption key. See Dataset encryption on page 969.
- Created at least two classifications. See Creating a classification on page 934.

About this task

You can delete a classification from your project any time after you create a classification. For example, you might want to delete the classification if the classification that was created earlier is not required.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification that you want to delete from the list under **Classification**.

   Note: Before deleting a classification associated with encrypted datasets, you must remove or move the encrypted datasets under it to another classification so that the delete icon is enabled.

5. Click the **Delete** icon .
   The **Delete Classification** dialog is displayed.
6. Click **Delete** to confirm the deletion of the classification.

Results

You deleted a classification.

Granting access to classification

You can grant or revoke access to the classification in your project to individual members with different roles or all members with a specific role. Without access to the classification, members cannot view, create, edit, delete, or use the classification.

Before you begin

You must be a *Project Owner* and have completed the following tasks:

- Created a project in Test. See Managing DevOps Test projects on page 1724.
- Added one or more users to your project. See Adding users to a project on page 1732.
- Created at least one classification. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification in the list under **Classification** for which you want to grant the access.
5. Choose any of the following methods to grant access to a member:
   Choose from:

   - Select the checkbox for the specific role under **Grant Read Only Access to roles** to grant read-only access to the members of a specific role. For example, if you select **Tester** then all members in the project with a tester role are granted read-only access to the selected classification. You can select any role or all the roles listed.
   - To grant read/write access to specific members, perform the following steps:
     a. Click **Add Members** under **Members with Read/Write Access**.

        The **Additional Members** dialog is displayed.

     b. Enter the name or email ID to search the user or navigate through the list and select the member you want to grant access. You can select multiple members in the list.

        Important:

        Irrespective of the role that the member (**Owner**, **Tester**, or **Viewer**) was assigned in the project, access to the classification has to be specifically granted to the members from the **Security** page.

     c. Click **Add Members.**

        You granted read/write access to the added members.
6. Choose any of the following methods to revoke access to a member:
   Choose from:

- Clear the checkbox for the specific role under **Grant Read Only Access to roles,** to revoke read-only access to the members of a specific role. For example, if you clear the **Tester** checkbox then access for all members in the project with a tester role is revoked for the selected classification. You can select any role or all the roles listed.
- Click the **Delete** icon in line with the member name that you want to remove the specific members from the **Members with Read/Write Access** list.

Notes:

- Any member with access to the classification can remove access to other members specifically added or of all members with a specific role.
- Members with access to the classification can remove themselves from the access list provided that there is at least one member in the list. After removing themselves, members cannot add themselves back to the access list and must be added by any of the other members in the list.

Results

You have granted or revoked the classification access to project members.

Management of database connections

You can add a database connection to a project and import the data from an external database in Test .

You can also restrict the database connection when you add it to a project and then add it to a classification to grant or restrict access for other members of the project. After you add a database connection you can create the database query that can be used to create test assets.

You can find the following information about adding the database connection and managing them in Test:

- Adding a database connection on page 938
- Adding a database connection to a classification on page 941
- Removing a database connection from a classification on page 942
- Editing a database connection on page 943
- Duplicating a database connection on page 944
- Deleting a database connection on page 946

Adding a database connection

You can add a database connection to your project in Test to import the data from an external database and create the test assets. You can restrict the project members from using this database connection by adding it to a Classification and providing permission only to the required members.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Ensured that you have a valid Java Database Connectivity (JDBC) URL and credentials for the database.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Infrastructure > Database Connections**.
   Result
   The **Database Connections** page is displayed.
4. Click **New Connection**.
   Result
   The **New Connection** dialog is displayed.
5. Perform the following actions in the **Details** tab:

   a. Enter a name for the database connection in the **Name** field.

   b. Select the database from the **Driver** list.
      The following are the supported databases:
      - Oracle
      - PostgreSQL
      - IBM DB2

   c. Enter the URL for the database in the **URL** field.

   d. Enter the username for the database in the **Username** field.

   e. Enter the password for the database in the **Password** field.

   f. Enter the schema name for the database in the **Schema** field.

   g. Add a description for the database connection in the **Description** field.

6. Select the **Restrict access using Classifications** checkbox if you want to restrict other members of the project from using this database connection.

   Later you can grant access to the required members by adding the connection to a classification.

   Note: The option, **Restrict access using Classifications** is not selected. This is the default selection.

7. Perform the following steps to add the SSL certificate for encrypting the data exchanged between Test and the database.

a. Click the **SSL** tab.

b. Click **Select File** and browse to the location on your computer where you have the Java Keystore File.

c. Select the file, and then click **Open**.

Result

The selected file name is displayed.

d. Enter the password for the Java Keystore File in the **Password** field.

8. Click **Test** to test the connection with the database.

Result

A message is displayed that confirms the connection with database.

9. Click **Save** to save the database.

A message is displayed that confirms database connection is added.

You can view the following details of the database connection on the **Database Connections** page:

Column header	Description	
Name	Displays the name of the database connection that you added.	
URL	Displays the URL of the database connection that you added.	
User	Displays the name of the user who added the database connection.	
Description	Displays the description that you entered when you created the database connection.	
Actions	or	Click the lock icon to go to the **Security** page. You can grant access to the database connection by adding it to a classification. **Note:** The unlock icon is displayed to indicate that the database connection is not restricted for other members of the project, and they can use database connection to create the test assets.
		Click the edit icon to edit the database connection. See Editing a database connection on page 943.
		Click the duplicate icon to duplicate the database connection. See Duplicating a database connection on page 944.
		Click the delete icon to delete the database connection. See Deleting a database connection on page 946.

Results

You have added a database connection to your project.

What to do next

You can add the database to a classification to provide access to other members of the project, if you have restricted it when you created the database connection. See Adding a database connection to a classification on page 941.

You can create a database query using the database connection that you added to your project. See Creating a database query on page 947.

Adding a database connection to a classification

You can add the database connection to a classification to grant access to the classification for other project members in Test . After the access is provided, they can use the database connection to create the database query and import the data from the database.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one database connection and restricted the access to other members of the project. See Adding a database connection on page 938
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created at least one classification. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   Result
   The **Security** page is displayed.
4. Click a classification for which you want to add the database connection in the list under **Classification**.
5. Click the **Add Connection** icon to select the database that you want to add to the classification.
   Result
   The **Add Connection** dialog is displayed.
6. Click the **Database Connection** field to view the available database connection in the project.
7. Identify and select the database connection that you want to add to the classification.
8. Enter the password that you use to connect to the database in the **Password** field.

Tip: You can click the view icon to view the entered password and ensure that you entered the correct password.

9. Click **Add**.
   Result
   The database connection that is added to the classification is displayed.

Results

You have added a database connection to the classification.

What to do next

You can create a database query by using the database connection that you added to your project. See Creating a database query on page 947.

Removing a database connection from a classification

You can remove a database connection from a classification when you are no longer required to provide access to other members of your project.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one restricted database connection to a classification. See Adding a database connection to a classification on page 941.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   Result
   The **Security** page is displayed.
4. Click a classification in the list under **Classification** from which you want to delete the database connection.
5. Click the **Delete** icon in line with the database connection that you want to delete from the classification.
   Result
   The **Remove database connection** dialog is displayed.

6. Click **Remove**.

Note: Removing the database connection from the classification also removes the password stored in Test for that database. You must enter the password again to gain access to the database connection while adding it to a different classification.

Results

You have removed the database connection from a classification.

Editing a database connection

You can edit the database connection, when you want to change the database details that you added earlier or if you want to update the SSL certificate file.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one database connection. See Adding a database connection on page 938
- Ensured that you have credentials for the database that you want to edit in a Test project.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Infrastructure > Database Connections**.
   Result
   The **Database Connections** page is displayed.
4. Click the Edit icon inline with the database that you want to edit.
   Result
   The **Edit connection** dialog is displayed.
5. Perform the following actions in the **Details** tab:

   a. Edit the details of the connection.

   b. Enter the password for the database in the **Password** field.

   c. Select the **Restrict access using Classifications** checkbox if you want to restrict other members of the project from using this database connection.

Later you can grant access to the required members by adding the connection to a classification.

Note: The option, **Restrict access using Classifications** is not select. This is the default selection.

6. Perform the following steps to change or add the SSL certificate and encrypt the data exchanged through the database.
    a. Click the **SSL** tab.
    b. Click the **x** icon to remove the existing Java Keystore file.
    c. Click **Select File** and browse to the location on your computer where you have the Java Keystore File.
    d. Select the file, and then click **Open**.
    Result
    The selected file name is displayed.
    e. Enter the password for the Java Keystore File in the **Password** field.
7. Click **Test** to test the connection with the database.
    Result
    A message is displayed that confirms the connection with database.
8. Click **Save** to save the database connection.

    A message is displayed that confirms database connection is modified and the edited database connection is displayed on the **Database Connections** page.

Results

You have edited the database connection in your project.

Duplicating a database connection

When you want create a similar database connection with different user credentials, to create the similar database connection and restrict or unrestricted it, or to create a similar database connection by encrypting the data exchanged between Test and database, you can create a new database connection using an existing database connection.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one database connection. See Adding a database connection on page 938.
- Ensured that you have a valid URL and credentials for the database that you want to add or duplicate in a Test project.

About this task

You can duplicate database connection and create a new one in the following scenarios:

- Creating the same database connection with or without restricting the access to other members of the project.
- Adding the same database connection to a different classification which consists of different datasets, secrets and have access to different group of people.
- Adding the same database with different user credentials.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Infrastructure > Database Connections**.
   Result
   The **Database Connections** page is displayed.
4. Click the **Duplicate** icon inline with the database connection that you want to duplicate.
   Result

   The **Duplicate connection** dialog is displayed.

   The value in the **URL** field, and the selections in the **Driver** field and the **Restrict access using Classifications** checkbox are auto-populated with the current values and selections in the database connection that you duplicate.
5. Perform the following actions in the **Details** tab:

   a. Enter a different name for the database connection in the **Name** field.

   b. Enter the username for the database in the **Username** field.

   c. Enter the password for the database in the **Password** field.

   d. Select the **Restrict access using Classifications** checkbox if you want to restrict other members of the project from using this database connection.

      Later you can grant access to the required members by adding the connection to a classification.

      Note: The option, **Restrict access using Classifications** is not select. This is the default selection.
6. Perform the following steps to change or add the SSL certificate for encrypting the data exchanged through the database.

   a. Click the **SSL** tab.

   b. Click the **x** icon to remove the existing Java Keystore file.

   c. Click **Select File** and browse to the location on your computer where you have the Java Keystore File.

d. Select the file, and then click **Open**.
Result
The selected file name is displayed.

e. Enter the password for the Java Keystore File in the **Password** field.

7. Click **Test** to test the connection with the database.
Result
A message is displayed that confirms the connection with database.
8. Click **Save** to save the database connection.
Result

A message is displayed that confirms database connection is added and the new database connection created by using the existing database connection is displayed on the **Database Connections** page.

Results
You have created a new database connection by using the existing database connection in your project.

Deleting a database connection

You can delete the database connection from a project when it is no longer required.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one database connection. See Adding a database connection on page 938.

1. Log in to Test.
The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

The **Overview** page of the project is displayed.

3. Click **Infrastructure > Database Connections**.
Result
The **Database Connections** page is displayed.
4. Click the **Delete** icon inline with the database connection that you want to delete.
Result
The **Delete Connection** dialog is displayed.
5. Click **Ok** to confirm the deletion.
Result
The **Connection deleted successfully** message is displayed.

Results

You have deleted a database connection from your project.

Management of database query

You can find information about creating, editing, duplicating, and deleting the database query in Test .

You can find the following information about creating and managing the database queries:

- Creating a database query on page 947
- Editing a database query on page 949
- Duplicating a database query on page 951
- Deleting a database query on page 952
- Publishing all changes in an Edit branch

Creating a database query

When you have the data that is useful for creating the test assets is stored in a database, you can import the data from the database and create a database Query in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Added at least one database connection. See Adding a database connection on page 938
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Database Query** on the **Test Editor** page.

- Click the **Action Menu** icon and select **Database Query**.
- Right-click the application in the Data navigator panel, select **New > Database Query**.

The **Edit Branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 7 on page 948.

6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

   Result

   The **Create database query** dialog is displayed.
7. In the **Create database query** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a dataset.
   b. Select **Location** from the drop-down list to store the dataset.
   c. Type a name for the database query.
   d. Click **Save**.

   Result

   The database query is created and displayed on the **Data** page. The database query created is listed under the `Database Queries` container within the selected application.
8. Perform the following steps in the **Data** page to select the table and query the data from the database:
   a. Select the database from the **Connection** list.
   b. Select the table in the selected database from the **Table** list.

      Result

      The data from the selected table is displayed.
   c. Select the required column of the table from which you want to import the data.
   d. Click **View** to switch between the **Visual** and **Code** view.

Note: Note: The default view is **Visual** view. In the **Code** view you can type the query to get data from the database.

e. Click the **Preview** icon to view the queried data from the database in tabular format in the bottom window.

f. Click **Save** icon to save the queried data from the database.

Results

You have created a database query in your project.

What to do next

Publish the database query to the Git repository so that other members of the project can use the database query. See Publishing all changes in an Edit branch.

You can create a dataset using the database query. See Creating a dataset by using a database query on page 960

You can create a Data definition using the database query. See Creating a data definition by importing fields from a database query on page 986.

Editing a database query

You can edit the database query when you want to query different data from the database or if you want to query the data from a different database.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created at least one database query. See Creating a database query on page 947.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.

5. Navigate to the **Database Queries** container, then right-click the database query you want to edit and select **Edit**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 950.

6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

7. Perform the following steps on the **Data** page to select a different table or query the different data from the same database or different database:

   a. Select a different table, modify the query, or different connection.

   b. Click **View** to switch between the **Visual** and **Code** view.

      Note: The default view is **Visual** view. In the **Code** view you can type the query to get data from the database.

   c. Click the **Preview** icon to view the queried data from the database in tabular format in the bottom window.

   d. Click the **Save** icon to save the queried data from the database.

Results

You have edited a database query in your project.

What to do next

You can publish the database query to the Git repository so that other members of the project can use the database query. See Publishing all changes in an Edit branch.

Duplicating a database query

You can create a new database query by using the existing database query, when you want the same database query with a different name or only slight modification required in the existing database query.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created at least one database query. See Creating a database query on page 947.

About this task

When you want to create a database query by using the existing database query, you must have an Edit branch. You can create the Edit branch during the creation of the database query. All your dataset resides in the Edit branch until you publish or discard the changes.

After you create a database query using the existing database query, you can either publish your changes to a remote Git repository or you can discard the changes made to the database query. The Edit branch automatically disappears from the **Branch** list as well as from the remote Git repository and sets the global branch to the previously selected branch in the following conditions:

- When you publish changes to the remote Git repository.
- When you discard the changes made in the dataset.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Navigate to the logical folder **Datasets** in the Data navigator, right click the database query that you want to save with a different name, and then click **Save As**.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

7. Enter a different name for the database query in the **Name** field.
8. Select a folder in the **Location** field.
9. Click **Save**.
   Result

   A message is displayed that confirms database query is created.

   The database query is displayed on the **Data** page. The dataset created is listed under the logical folder `Datasets` under the selected physical folder that you selected. The name of the database query is suffixed with the `.dtd.yaml` file extension.
10. Make any required changes to the existing query, and then click **Save** icon to save the queried data from the database.

Results
You have created a database query using the existing database query in your project.

What to do next

You can publish the database query to the Git repository so that other members of the project can use the database query. See Publishing all changes in an Edit branch.

Deleting a database query

You can delete a database query when it is no longer required.

Before you begin
You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created at least one database query. See Creating a database query on page 947.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.

3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.

4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.

5. Perform one of the following actions:
   a. Select the database query to open it on the **Data** page. Click the **Delete** icon .
   b. Right-click the database query you want delete and select **Delete**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 953.

6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

   Result

   The **Delete database query** dialog is displayed.

7. Click **Delete** to confirm the deletion.

   Result

   A message is displayed that confirms database query is deleted.

Results

You have deleted the selected database query from your project repository.

What to do next

You can publish the database query to the Git repository so that other members of the project can use the database query. See Publishing all changes in an Edit branch.

Management of datasets

You can find information about creating datasets in Test , applying a dataset classification, editing datasets, and managing datasets that are in your project repository.

A dataset contains variable data that are used by a test at test run time. You create the datasets and associate them with your tests in the desktop products. You can decide whether you want to use the dataset associated with the test to be used at run time or override that associated dataset with another dataset at run time. You can create datasets in the desktop products itself and select the datasets as overrides in the desktop products.

When you want to run those tests from Test and you want the tests to use a different set of variable data, you can create datasets in Test. You can then select them to override the dataset associated with the tests that are in the project repository.

You can create a dataset in Test by using any of the following methods:

- Creating an empty dataset. See Creating a dataset on page 954.
- Importing data from a CSV file. See Creating a dataset by using a CSV file on page 956.
- Creating a dataset with fabricated data. See Creating a dataset by using a Data definition on page 958.

When you want to edit the data in a dataset, you must open it for editing. See Editing a dataset on page 962.

After you edit the dataset, you must publish the changes, so that the dataset is added to the project repository. See Publishing changes in datasets.

If the data in your dataset contains sensitive information, you can encrypt the dataset columns. See Encrypting a dataset column on page 969.

If a dataset has encrypted columns and you want to view an encrypted dataset, see Viewing an encrypted dataset on page 975.

When you want to remove the encryption for dataset columns, you can decrypt the dataset column. See Decrypting a dataset column on page 971.

You can assign a classification for encrypted datasets. See Managing an encrypted dataset on page 976.

You can change the classification of an encrypted dataset. See Changing classification for an encrypted dataset on page 976.

Creating a dataset

You can create datasets in Test to replace the values of existing dataset with the values of new dataset during a test or schedule run.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Dataset** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Dataset**.
   - Right-click the application in the data navigator panel, select **New > Dataset**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to 7 on page 956.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

Result

The **Create dataset** dialog is displayed.

7. In **Step 1** of the **Create dataset** dialog, perform the following tasks:
    a. Select the **Application** for which you want to create a dataset.
    b. Select **Location** from the drop-down list to store the dataset.
    c. Type a name for the dataset.
    d. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create dataset** dialog, perform the following tasks:
    a. Select **Source** as **Create an Empty dataset**.
    b. Click **Create**.

    Result

    The dataset is displayed on the **Data** page. The dataset created is listed under the logical folder `Datasets` within the selected application.
9. Enter the variable data in the table to match the data that you want to use in tests.
10. Click the **Save** icon to save the dataset.

Results

You created a dataset in your project.

What to do next

You can perform the following tasks:

- Add, modify, or remove data in the dataset. See Editing a dataset on page 962.
- Publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing changes in datasets.

Related information

Creating a dataset by using a CSV file on page 956

Creating a dataset by using a Data definition on page 958

Creating a dataset by using a CSV file

When you use variable data that is contained in a CSV file, you can import the data from the CSV file and create a dataset in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Saved a CSV file that contains variable data on your computer.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Dataset** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Dataset**.
   - Right-click the application in the data navigator panel, select **New > Dataset**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 958.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

Result

The **Create dataset** dialog is displayed.

7. In **Step 1** of the **Create dataset** dialog, perform the following tasks:
    a. Select the **Application** for which you want to create a dataset.
    b. Select **Location** from the drop-down list to store the dataset.
    c. Type a name for the dataset.
    d. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create dataset** dialog, perform the following tasks:
    a. Select **Source** as **Import a CSV file**.
    b. Click **Select File** and choose the CSV file that contains variable data to import into the dataset.
    c. **Toggle on** the **First row contains headers** option if your CSV file includes column headers.
    d. Click **Create**.

Results

You have created the dataset by using the CSV file in your project.

What to do next

You can perform the following tasks:

- Add, modify, or remove data in the dataset. See Editing a dataset on page 962.
- Publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing changes in datasets.

Related information

Creating a dataset on page 954

Creating a dataset by using a Data definition on page 958

Creating a dataset by using a Data definition

When you created a Data definition in Test to fabricate test data, you can create a dataset by using the Data definition.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

- Created a data definition. See Creating an empty data definition on page 980.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Dataset** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Dataset**.
   - Right-click the application in the data navigator panel, select **New > Dataset**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 959.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

   Result

   The **Create dataset** dialog is displayed.
7. In **Step 1** of the **Create dataset** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a dataset.
   b. Select **Location** from the drop-down list to store the dataset.

c. Type a name for the dataset.
d. Click **Next** to proceed to Step 2.

8. In **Step 2** of the **Create dataset** dialog, perform the following tasks:
   a. Select **Source** as **Fabricate data**.
   b. Select the Data definition to use from the **Select Data Definition** list.
   c. Enter the number of rows in the **Number of Rows** field.
   d. Enter a seed value in the **Seed** field.
   e. Click the **Fabricate headers** toggle to generate a header row.
   f. Click **Create**.

Results

You have created the dataset by using a Data definition in your project.

What to do next

You can perform the following tasks:

- Add, modify, or remove data in the dataset. See Editing a dataset on page 962.
- Publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing changes in datasets.

Related information

Creating a dataset on page 954

Creating a dataset by using a CSV file on page 956

Creating a dataset by using a database query

When you use variable data that is contained in a database, you can create a database query to import the required data from a database and use the data that is present in the database query to create a dataset in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created at least one database query. See Creating a database query on page 947.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Dataset** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Dataset**.
   - Right-click the application in the data navigator panel, select **New > Dataset**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 961.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

   Result

   The **Create dataset** dialog is displayed.
7. In **Step 1** of the **Create dataset** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a dataset.
   b. Select **Location** from the drop-down list to store the dataset.
   c. Type a name for the dataset.
   d. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create dataset** dialog, perform the following tasks:

a. Select **Source** as **Use a Database Query**.
b. Select the database query to use from the **Select Database Query** list.
c. Enter the number of rows in the **Number of Rows** field.
d. Enter a seed value in the **Seed** field.
e. Click **Create**.

Results

You have created the dataset by using a database query in your project.

What to do next

You can perform the following tasks:

- Add, modify, or remove data in the dataset. See Editing a dataset on page 962.
- Publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing changes in datasets.

Related information

Creating a dataset on page 954

Creating a dataset by using a CSV file on page 956

Editing a dataset

You can add, modify, remove, import, or export data from a dataset.

Before you begin

You must have created a dataset or configured a repository that contains the dataset.

About this task

If you are a project **Owner** or **Tester** in Test , you can perform basic tasks in the datasets by right-clicking any row, column, or cell in the dataset to organize your data in a better way. For example, you can perform tasks such as updating data in a cell, inserting or deleting rows and columns, or renaming column names.

After you edit the dataset, you can save the changes made to the dataset, and then you can publish the dataset to the Git repository. If you save and close the edited dataset, the **Changes** page lists the edited dataset and later you can publish it to the Git repository for other members to use.

When you have a CSV file that has data separated from a character, then you can import that CSV file into the dataset. You can select any of the following separator characters from the **Configure Dataset** window, and the selection can be the separator character that you used in the CSV file:

- Comma
- Semicolon
- Space
- Tab
- Other

Consider that you have the data in the CSV file in the following format:

Name;CCNum
John;1234 5678 1234 5678
Bob;1122 3344 5566 7788
Amy;2233 4455 6677 8899

When you import the CSV file in the dataset, and then select the separator value as **Semicolon**, the data in the dataset is displayed as follows:

	Name	CCNum
1	John	1234 5678 1234 5678
2	Bob	1122 3344 5566 7788
3	Amy	2233 4455 6677 8899

If you want the data in its original format, that is, a semicolon (;) character to separate the data, then you can choose any other separator value from the **Configure Dataset** dialog.

Note: The default separator value is **Comma**.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch that contains your project with the dataset that you want to edit.

   The projects in the selected branch are displayed in the Data navigator panel. The dataset are listed under the folder you created to contain the dataset within a logical folder named `Datasets`.

5. Click the dataset in the data navigator.

   The dataset opens in the right pane.

6. Click **Edit**.

   Alternatively, right-click the datset in the data navigator panel, and then click **Edit**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 8 on page 964.

7. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

     Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

8. Perform the following actions to use the options available for editing:

Options	Actions
Find and Replace	To find: a. Click the **Find and Replace** icon. b. Enter the content that you want to search in the **Find** field. c. Select any or all the following options to find the search content more effectively: ▪ Select the **Case sensitive** icon to search the content that is the exact letter case of the content entered in the **Find** field. ▪ Select the **Match entire cell contents** icon to search for cells that contain only the characters that you have entered in the **Find** field. ▪ Select the **Search using regular expression** icon to search the pattern that matches strings. For example, to search a cell that contains any number between 0 to 9, do the following:

Options	Actions
	i. Enter \d in the **Find** field. ii. Select the **Search using regular expression** checkbox. iii. Click **Find**. d. Click the **Find** icon. If the text is found, the cell containing that text is selected. e. Click the **Find** icon again to find further instances of the search text. To find and replace: a. Click the **Find and Replace** icon. b. Enter the content that you want to search in the **Find** field. c. Enter the content that you want to replace in the **Replace** field. d. Select any or all the following options to find and replace the content more effectively: ▪ Select the **Case sensitive** icon to find the content that is the exact letter case of the content entered in the **Find** field. ▪ Select the **Match entire cell contents** icon to find and replace for cells that contain only the characters that you have entered in the **Find** and **Replace** fields. ▪ Select the **Search using regular expression** icon to find and replace the pattern that matches strings. e. Click the **Replace** icon to replace the individual instances. f. Click the **Replace All** icon to replace every instance of the content throughout the dataset.
Undo	a. Click the **Undo** icon. b. Select the recent changes from the list that you want to undo, and then click the list. The **Undo** option reverts the change you made in the dataset.
Redo	a. Click the **Redo** icon. b. Select the recent changes from the list that you want to redo, and then click the list.
Import	You can import variable data into a dataset from the following sources: ◦ When there is a large amount of data stored in a CSV file. ◦ When you use the data fabrication feature to generate test data. **Importing data from a CSV file**

Options	Actions
	a. Click the **Import** icon . b. Select **Import a CSV file** from the drop-down list. c. Click **Select file** and choose the CSV file that contains variable data to import into the dataset. d. Enable the **First row contains headers** by using the toggle icon if your CSV file contains the header. e. Select one of the following options to append or overwrite data in the dataset: ▪ Enable **Overwrite existing data** by using the toggle icon to add the rows and columns from the beginning of the dataset. ▪ Enable **Append existing data** by using the toggle icon to add rows and columns to the end of the dataset. f. Click **Import**. **Importing data by using data fabrication** a. Click the **Import** icon . b. Select **Fabricate data** from the drop-down list. c. Select a Data definition from the drop-down list. d. Enter the number of rows that you want to generate in the **Number of Rows** field. e. Enter a seed value in the **Seed (optional)** field. f. Enable **Fabricate headers** by using the toggle icon if your schema contains the headers. g. Select one of the following options to append or overwrite data in the dataset: ▪ Enable **Overwrite existing data** by using the toggle icon to add the rows and columns from the beginning of the dataset. ▪ Enable **Append existing data** by using the toggle icon to add rows and columns to the end of the dataset. h. Click **Import**.
Export	You can export variable data from the dataset into a CSV file to reuse in future tests when required. You must have opened a dataset from which you want to export the data.

Options	Actions
	Click the **Export** icon to download the dataset as a CSV file.
Set as current row	During the test run, if you want variable data to be selected from a current row instead of the first row in a dataset, right-click any cell in a row and select **Set as current row**. **When rows are deleted:** If you delete any row between row 1 and the current row, the current row data is taken from the next row. For example, when you set the current row as 6, and then you delete any row between row 1 to row 6, the current row remains at row 6, but the content of row 7 is moved to row 6. **When rows are inserted:** If you insert any new row between row 1 to the current row, the current row data is taken from the previous row. For example, when you set the current row as 6, and then you insert any row between row 1 to row 6, the current row remains at row 6, but the content of row 5 is moved to row 6.
Dataset Configuration	You can configure the dataset by clicking the **Menu** icon, and then selecting **Configure**. In the **Configure Dataset** dialog, you can change the row and column settings and configure the string values in the dataset that contains variable data for tests to use when they run. ◦ Select any of the separator values that you used in the CSV file. The available options are **Comma**, **Semicolon**, **Space**, **Tab**, and **Other**. In the CSV file, if you have any other separator characters other than the available options, then you can select the **Other** option, and then can specify a value. For example, if the data in the CSV file is separated by a character #, then select the **Other** option and enter # in the field. ◦ Configure the following options to change the row and column settings: ▪ **Column header** - Use an up-down control button to increment or decrement the value of the column header. ▪ **Data start point** - Use an up-down control button to increment or decrement the value of the data starting pointer. ▪ **Current row** - Use an up-down control button to increment or decrement the value of the current row. ◦ Configure the following options to change the string values in the dataset:

Options	Actions
	▪ **Treat as null** - Enter a string value that is to be treated as null when running the test. ▪ **Treat as empty** - Enter a string value that is to be treated as empty when running the test. For example, when you run the test and the data *123* in the dataset is to be treated as empty, then you must specify *123* in the **Treat as empty** field. ▪ **Treat empty text as null** - Select this field when you want the dataset that contains any blank cells, and the value of those blank cells to be interpreted as null. ◦ Click **Update** to apply the changes.
Discard	Click the **Menu** icon and select **Discard** to discard the changes made to the dataset.

9. Click the **Save** icon to save the changes made to the dataset or the **Save All** icon in the Data navigator panel to save all changes made to different datasets by using the same Edit branch.

Note: When editing multiple assets (such as datasets, data definitions or schemas) simultaneously, right-click a test asset in the Data page to manage open tabs using the following options:

To...	Do this...
Keep your current file open and close the open files on the right.	Right-click, and then click the **Close Tabs to right** option.
Keep your current file open and close the open files on the left.	Right-click, and then click the **Close Tabs to left** option.
Keep your current file open and close all the other open files.	Right-click, and then click the **Close Others** option.
Close all open file tabs.	Right-click, and then click the **Close All Tabs** option.

10. Click the **Publish** icon to publish the dataset to the Git repository.

Results

You have edited the dataset.

Dataset encryption

Encrypted datasets are useful when you want to run tests that contain confidential information such as a set of passwords or account numbers.

When you run a test in Test that uses an encrypted dataset, then you must provide an encryption key for decrypting the encrypted data in columns so that the data can be used in the test. If the test uses data from the multiple encrypted dataset columns, you must enter the same encryption key for every encrypted dataset column that the test uses.

When you run the test that uses the dataset with the encrypted column, the value of the column is decrypted at a run time. The data in the column is sent as a clear-text string in requests to the server. The actual values of the encrypted dataset variables are not displayed in the test log. The test log displays asterisks for the encrypted dataset variables.

You can use only one encryption key to encrypt data in the columns in any dataset.

You can set the dataset classification (encrypted password) for a dataset. The dataset classification that you set is applied for the dataset if it is saved in different branches of the project repository and you need not set a password for the dataset in each branch. If you want to use different passwords, then you can create a different dataset by cloning the dataset that you want.

Important:

The encryption keys that you use to encrypt data in a dataset are not stored on the server, nor they can be retrieved from the server. Therefore, you must remember to store the encryption keys in a secure location. You must use the same encryption keys to perform the following operations:

- View the encrypted values
- Decrypt data
- Enable the use of the encrypted dataset during test runs

Encrypting a dataset column

To secure test data, you must encrypt datasets in Test . You can encrypt data in the columns of a dataset by using an encryption key. When you run a test that utilizes a dataset with encrypted variables, you must enter the encryption key for the encrypted column that the test uses.

Before you begin

You must have created a dataset. See Creating a dataset on page 954.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

The **Overview** page of the project is displayed.

3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.

4. Select the branch that contains your project with the dataset that you want to edit.

   The projects in the selected branch are displayed in the Data navigator panel. The dataset are listed under the folder you created to contain the dataset within a logical folder named `Datasets`.

5. Click the dataset in the data navigator.

   The dataset opens in the right pane.

6. Click **Edit**.

   Alternatively, right-click the datset in the data navigator panel, and then click **Edit**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 8 on page 964.

7. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

8. Right-click any cell in a column that you want to encrypt and select **Encrypt column data**.
   Result
   The **Encrypt Column** window is displayed.
9. Enter an encryption key in the **Encryption Key** field to encrypt the data in the column.

   Remember: When you have already encrypted other columns in the dataset, you must enter the same encryption key that you used previously. You can use only one encryption key to encrypt columns in a dataset.

   Important:

The encryption keys that you use to encrypt data in a dataset are not stored on the server, nor they can be retrieved from the server. Therefore, you must remember to store the encryption keys in a secure location. You must use the same encryption keys to perform the following operations:

- View the encrypted values
- Decrypt data
- Enable the use of the encrypted dataset during test runs

10. Click **Encrypt Column**.
    Result
    Asterisks are displayed instead of actual data for the encrypted column.
11. Click the **Save** icon to save the changes to the dataset.

Results

You have encrypted the dataset column in your project.

What to do next

You can publish the dataset to the Git repository so that other members of the project can use the dataset by clicking the **Publish** icon. See Publishing changes in datasets.

Decrypting a dataset column

To view the content of an encrypted dataset in Test , you can decrypt the dataset column. Removing encryption from a dataset revokes the protection offered to the test data.

Before you begin

You must have created at least one dataset and encrypted the dataset with an encryption key. See Creating a dataset on page 954 and Encrypting a dataset column on page 969.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch that contains your project with the dataset that you want to edit.

   The projects in the selected branch are displayed in the Data navigator panel. The dataset are listed under the folder you created to contain the dataset within a logical folder named `Datasets`.
5. Click the dataset in the data navigator.

The dataset opens in the right pane.

6. Click **Edit**.

   Alternatively, right-click the datset in the data navigator panel, and then click **Edit**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 8 on page 964.

7. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

8. Right-click encrypted cells that display the contents with asterisks, and then select **Decrypt column data**.
   Result
   The **Decrypt Column** window is displayed.
9. Enter the encryption key that you used to encrypt the data in the column in the **Encryption Key** field.
10. Click **Decrypt Column**.
    Result
    Asterisks are replaced with the actual data in the decrypted column.
11. Click the **Save** icon to save the changes to the dataset.

Results

The encryption is removed from the selected column in the dataset. When you run a test that uses a dataset that contains decrypted data, the variable data is substituted for the original data in the recorded test without prompting for the encryption.

What to do next

You can publish the dataset to the Git repository so that other members of the project can use the dataset by clicking the **Publish** icon. See Publishing changes in datasets.

Viewing a dataset

When test assets includes datasets, you can view the contents of a dataset from Test . The datasets residing in the Git repository are listed in the **Datasets** page.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created or joined a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created at least one dataset and encrypted the dataset with an encryption key. See Creating a dataset on page 954 and Encrypting a dataset column on page 969.
- Created a classification for an encrypted dataset. See Managing an encrypted dataset on page 976.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch that contains your project with the dataset that you want to edit.

   The projects in the selected branch are displayed in the Data navigator panel. The dataset are listed under the folder you created to contain the dataset within a logical folder named `Datasets`.
5. Click the dataset in the data navigator.

   The dataset opens in the right pane.

Results

You have viewed the contents of the dataset.

Deleting a dataset

You can delete the dataset when it is not required in your test environment.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created at least one dataset in your Git repository that you have configured.

About this task

You can delete datasets in Test in the following scenarios:

- Datasets that you created in the desktop clients and that are cloned to the repository in your project on Test.
- Datasets that you created in Test and that are published to the repository in your project.
- Datasets that are in the `.csv` file format.

You cannot delete the following datasets:

- Datasets that you created or edited in Test and that are not published to the repository in your project.
- Datasets that are in the `.sit` file format.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch that contains your project with the datasets that you want to edit.

   The projects in the selected branch are displayed in the Data navigator panel. The datasets are listed under the folder you created to contain the dataset within a logical folder named as `Datasets`.
5. Right-click the dataset in the data navigator, and then click the **Delete** option.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 975.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

◦ Perform the following steps to create the Edit branch:

a. Select **Create new edit branch**.

b. Enter a name for the Edit branch in the **Branch Name** field.

c. Click **Save**.

7. Perform the following steps in the **Delete <Dataset>** dialog that is displayed, where *<Dataset>* displays the name of the selected dataset.

a. Select the option to publish the deletion from the edit branch to the main branch.

b. Click **Delete**.

The selected dataset is deleted and the deletion is published from the edit branch to the main branch.

Results

You deleted a dataset.

Viewing an encrypted dataset

You can use the **Dataset** page to view the contents of an encrypted dataset from Test .

Before you begin

You must have opened the dataset to view the contents.

1. Perform the following steps, if you have created a classification but yet to add the encrypted dataset to it:

a. Find the encrypted dataset that you are interested in, and then right-click it.

b. Click the **Security** option in the menu.
The **Change the classification for the dataset** dialog is displayed.

c. Select a classification from the list and enter the encryption key set for the dataset.

d. Click **Save** to save the classification details.

2. Right-click the encrypted column and click **Show encrypted data**.
You can right-click the decrypted column and select **Hide encrypted data** to encrypt the data again.

Important: The **Show encrypted data** option is available only when you have added the encrypted dataset to the respective classification.

Results

You have viewed the contents of an encrypted dataset.

Changing classification for an encrypted dataset

To move the encrypted dataset from one classification to another dataset in Test , you can change the classification for an encrypted dataset.

Before you begin

You must have created at least two or more classifications. See Creating a classification on page 934.

1. Go to the **Datasets** page and find an encrypted dataset for which you want to change the classification.
2. Click the **Lock** icon from the **Actions** column of a dataset.
3. In the **Change the classification for the Dataset** window:
    a. Choose the classification from the list.
    b. Enter the encryption key for the dataset that was set while encrypting the dataset column.
    c. **Save** the classification details.

Remember: You can perform this task only if you know the encryption key set for the dataset while encrypting the dataset column.

Results

You have changed the classification for an encrypted dataset.

Related information

Managing an encrypted dataset on page 976

Creating a classification on page 934

Managing an encrypted dataset

You can use encrypted datasets to limit access to confidential information such as account number or passwords. You can arrange data by an appropriate category so that project members can use datasets more effectively in certain tests and protect them.

About this task

A dataset can contain classified information that other members can access with permission. As a project owner, you can group encrypted datasets into different classifications and enable project members to view and edit datasets and run tests associated with the encrypted datasets. After you created a classification, you can change the classification for a dataset. You can also delete a classification if you do not require it in your test environment.

Notes:

- You must grant access and provide an encryption key of an encrypted dataset to other members of the project to work with the encrypted dataset.
- A project member who has been added as a *Tester* role can work with the encrypted dataset.

Adding an encrypted dataset to a classification

As a project *Project Owner* or *Tester*, you can organize encrypted datasets by adding it to a classification so that project members can use and protect datasets more efficiently.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created a project in Test. See Managing DevOps Test projects on page 1724.
- Configured the repository that contains the test assets in your project. See Adding repositories to a server project on page 1729.
- Created at least one dataset and encrypted the dataset with an encryption key. See Dataset encryption on page 969.
- Created at least one classification. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click **Add Dataset** to select the encrypted dataset to become part of the classification.
   The **Add Dataset** dialog is displayed.
5. Perform the following steps:
   a. Click the **Encrypted dataset** field to view the encrypted dataset.
   b. Identify and select the encrypted dataset that you wanted to add to the classification.
   c. Enter the encryption key that you used to encrypt the dataset in the **Password** field.

      Tip: You can click the view icon to view the entered encryption key and ensure that you entered the correct key.

   d. Click **Add**.

      The encrypted dataset that is added to the classification is displayed.

   e. You can add other encrypted datasets to the same classification, if required.

What to do next

You can grant access to other project members to use the encrypted dataset. See Granting access to classification on page 936.

Moving an encrypted dataset to another classification

When you add many encrypted datasets to the same classification, you can move some of them to another classification or when you want to delete a classification but you want to retain the encrypted datasets associated with it, you can move them to different classification before deleting the classification.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created at least one dataset and encrypted the dataset with an encryption key. See Dataset encryption on page 969
- Created at least two classifications. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification from the list under **Classification** from which you want to move the encrypted dataset.
5. Click the **Edit** icon inline with the encrypted dataset that you want to move to another classification.
   The **Change the classification for the dataset** dialog is displayed.
6. Click a classification from the list under **Classification** to which you want to move the encrypted dataset.
7. Enter the encryption key that you used to encrypt the dataset in the **Password** field.
8. Click **Save**.

   The encrypted dataset is moved to the classification that you selected and the same is displayed on the **Security** page.

Results

You have moved the encrypted dataset from one classification to another.

Removing a dataset from the classification

You can remove a dataset from a classification when it is no longer required.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Created at least one dataset and encrypted the dataset with an encryption key. See Dataset encryption on page 969
- Created a classification and added the encrypted dataset to it. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification from the list under **Classification** from which you want to delete the encrypted dataset.
5. Click the **Delete** icon in line with the encrypted dataset that you want to delete from the classification.
   The **Remove Dataset** dialog is displayed.
6. Click **Remove**.

Note: Removing dataset from the classification also removes the password stored in Test for encrypted data. You must provide the password again while adding it to a different classification.

Results

You have removed the datasets from a classification.

Management of data definition

You can find information about creating, editing, and managing data definitions in Test , to support realistic and reusable test data generation.

You can create a data definition in Test by using any of the following methods:

- Creating an empty data definition. See Creating an empty data definition on page 980.
- Importing data from a CSV file. See Creating a data definition by importing fields from a CSV file on page 982.
- Creating a data definition from a dataset. See Creating a data definition by importing fields from a dataset on page 984
- Creating a data definition from a database query. See Creating a data definition by importing fields from a database query on page 986.

When you want to edit the data in a data definition, you must open it for editing. See Modifying a data definition on page 991.

After you edit the data fefinition, you must publish the changes, so that the data definition is added to the project repository. See Publishing changes in datasets.

Creating an empty data definition

You can create an empty data definition in Test as the first step to create a data definition. After you save the empty data definition within the selected application in your project repository, you can edit the data definition to add the different generators.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Read information about using data definitions to fabricate test data. See Overview of data fabrication.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:

   - Click the **Open action menu** icon in the Data navigator, and then click **Data definition**.
   - Click **Add files** in the right panel, and then click **Data definition**.
   - Right-click the application in the data navigator panel, select **New > Data definition**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 981.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

7. In **Step 1** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a data definition.
   b. Select **Location** from the drop-down list to store the dataset.
   c. Type a **Name** for the data definition.
   d. Add a description about the data definition in the **Description** field.
   e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select **Source** as **Create an empty data definition**.
   b. Click **Create**.

The new data definition opens in the **Data** page in the editing mode. The data definition created is listed under the logical folder `Data definitions` under the application selected.

Results

You created an empty data definition in your project.

What to do next

You can perform the following tasks:

- Add basic generators with their default settings to the data definition. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their default settings on page 988.
- Add basic generators to the data definition, and then modify the configuration of the generators. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their modified settings on page 990.

Related information

Creating a data definition by importing fields from a CSV file on page 982

Creating a dataset by using a Data definition on page 958

Creating a data definition by importing fields from a CSV file

When you want to create a data definition in Test with specific fields for the test data, you can import a CSV file that defines the specific fields. After you import fields to the data definition, you can edit the data definition and add generators for the specific fields before you generate the test data.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that a CSV file is available on your computer, from which you want to import data.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:

   - Click the **Open action menu** icon in the data navigator, and then click **Data definition**.
   - Click **Add files** in the right panel, and then click **Data definition**.
   - Right-click the application in the data navigator panel, select **New > Data definition**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 983.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

7. In **Step 1** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a data definition.
   b. Select **Location** from the drop-down list to store the dataset.
   c. Type a **Name** for the data definition.
   d. Add a description about the data definition in the **Description** field.
   e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select **Source** as **Imports fields from a CSV file**.
   b. Click **Select file** and browse to the location on your computer where you have the CSV file.
   c. Click **Create**.

The new data definition opens in the **Data** page in the editing mode. The data definition created is listed under the logical folder `Data definitions` under the application selected.

Results

You created a data definition in your project, which has the fields from the CSV added as the fields of the new data definition but no generators are added.

What to do next

You can perform the following tasks:

- Add basic generators with their default settings to the data definition. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their default settings on page 988.
- Add basic generators to the data definition, and then modify the configuration of the generators. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their modified settings on page 990.

Related information

Creating an empty data definition on page 980

Creating a dataset by using a Data definition on page 958

Creating a data definition by importing fields from a dataset

When you want to create a data definition in Test with specific fields for the test data, you can import the fields from a dataset. After you import fields to the Data definition, you can edit the data definition and add generators for the specific fields before you generate the test data.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that a dataset is available in your project repository, from which you want to import data.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:

   - Click the **Open action menu** icon in the Data navigator, and then click **Data definition**.
   - Click **Add files** in the right panel, and then click **Data definition**.
   - Right-click the application in the data navigator panel, select **New > Data definition**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 985.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

7. In **Step 1** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a data definition.
   b. Select **Location** from the drop-down list to store the dataset.
   c. Type a **Name** for the data definition.
   d. Add a description about the data definition in the **Description** field.
   e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create Data Definition** dialog, perform the following tasks:
   a. Select **Source** as **Import fields from a dataset**.
   b. Search for the dataset in your project repository by entering the file name in the **Dataset** field.

      Alternatively, browse to the location in your project repository, and then select the file.

   c. Click **Create**.

The new data definition opens in the **Data** page in the editing mode. The headers in the dataset file are added as the fields of the new data definition but no generators are added.The data definition created is listed under the logical folder `Data definitions` under the application selected.

Results

You created a data definition in your project, which has the fields from the dataset added as the fields of the new data definition but no generators are added.

What to do next

You can perform the following tasks:

- Add basic generators with their default settings to the data definition. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their default settings on page 988.
- Add basic generators to the data definition, and then modify the configuration of the generators. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their modified settings on page 990.

Related information

Creating an empty data definition on page 980

Creating a dataset by using a Data definition on page 958

Creating a data definition by importing fields from a database query

You can create data definition to generate a random test data that is required to run a test in Test using the specific database fields.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.
- Created at least one database query. See Creating a database query on page 947.

About this task

You must first create a database query and then create a data definition using the database query to import the specific fields from the database query. After you import fields to the data definition, you can edit the data definition and add generators for the specific fields before you generate the test data.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click the **Open action menu** icon in the data navigator, and then click **Data definition**.
   - Click **Add files** in the right panel, and then click **Data definition**.
   - Right-click the application in the data navigator panel, select **New > Data definition**.

The **Edit Branch** dialog is displayed.

If the **Edit branch** dialog is not displayed, then go to step 7 on page 987.

6. Select your action to either select an existing Edit branch or create an Edit branch:
    - Perform the following steps to select an existing Edit branch:
        a. Select **Use an existing edit branch**.
        b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

        c. Click **Ok**.
    - Perform the following steps to create the Edit branch:
        a. Select **Create new edit branch**.
        b. Enter a name for the Edit branch in the **Branch Name** field.
        c. Click **Save**.
7. In **Step 1** of the **Create Data Definition** dialog, perform the following tasks:
    a. Select the **Application** for which you want to create a data definition.
    b. Select **Location** from the drop-down list to store the dataset.
    c. Type a **Name** for the data definition.
    d. Add a description about the data definition in the **Description** field.
    e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create Data Definition** dialog, perform the following tasks:
    a. Select **Source** as **Import fields from a database query**.
    b. Select the required database query file in your project.
    c. Click **Create**.

The new data definition opens in the **Data** page in the editing mode. The columns in the database query are added as the fields of the new Data definition but no generators are added. The data definition created is listed under the logical folder `Data definitions` under the application selected..

Results

You have created a data definition in your project by importing fields from a database query.

What to do next

You can perform the following tasks:

- Add basic generators with their default settings to the data definition. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their default settings on page 988.
- Add basic generators to the data definition, and then modify the configuration of the generators. You can then save and publish the data definition. See Saving a data definition by adding basic generators with their modified settings on page 990.

Related information

Creating an empty data definition on page 980

Creating a dataset by using a Data definition on page 958

Saving a data definition by adding basic generators with their default settings

After you create a data definition in Test , you must add the basic generators with their default settings and save the data definition before you can generate test data.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an empty data definition that is opened for editing.

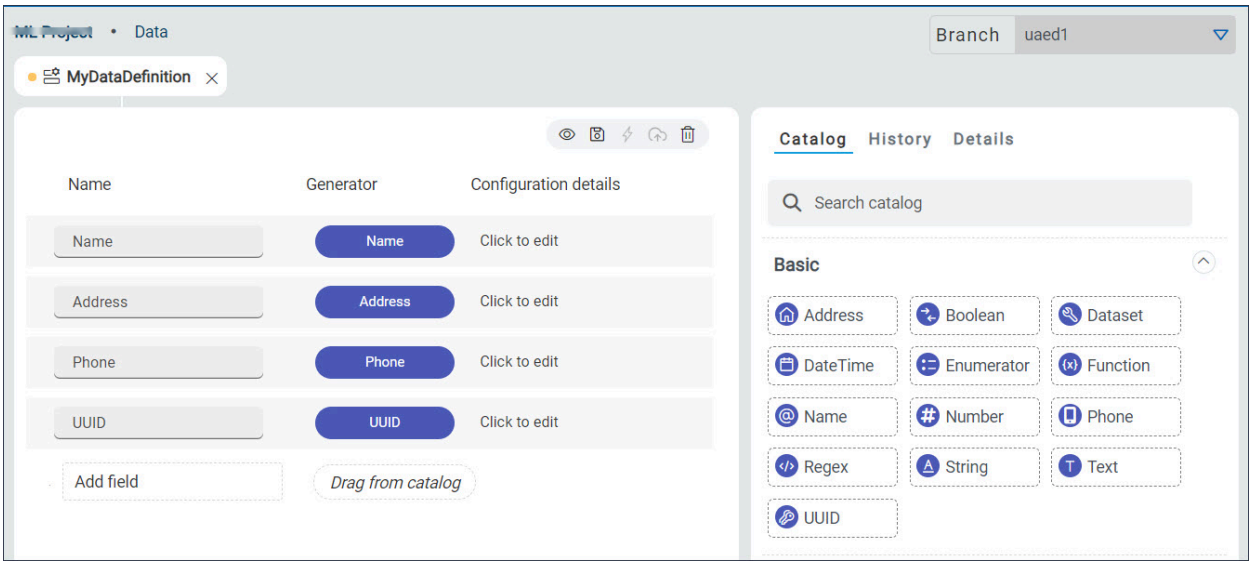
1. Drag a basic generator from the **Catalog** panel to the data definition.

   The basic generator is added as a field in the data definition.

2. Drag other basic generators from the **Catalog** panel to the data definition to complete the data structure that you want in the fabricated data.

   Do not change any of the fields in the data definition or the configuration of the basic generators.

   For example, if you want to fabricate data about persons with their name, address, phone number, and a unique ID for each name or record, you can drag the **Name**, **Address**, **Phone**, and **UUID** generators to the data definition, without making any changes to any of the fields in the data definition or in the basic generators.



3. Click the **Save** icon to save the changes made to the Data definition or the **Save All** icon to save all changes made to different Data definitions by using the same Edit branch.

Note: When you edit different files that include datasets or Data definitions simultaneously, you can select your action from the following options:

To...	Do this...
Keep your current file open and close the open files on the right.	Right-click, and then click the **Close Tabs to right** option.
Keep your current file open and close the open files on the left.	Right-click, and then click the **Close Tabs to left** option.
Keep your current file open and close all the other open files.	Right-click, and then click the **Close Others** option.
Close all open file tabs.	Right-click, and then click the **Close All Tabs** option.

Note: The changes you made are only visible and available to you in the Edit branch. You must publish the changes in the Edit branch to the branch in the project repository for the changes to be visible to the other members of the project.

4. Click the **Publish** icon to publish the changes made to the data definition to the branch in the project repository.

Results

You edited and saved a data definition after adding basic generators with their default settings.

What to do next

You can perform any of the following tasks:

- Preview the test data generated by using the generators added as fields in the data definition by clicking the **Preview** icon .
- Generate the test data by using the generators added as fields in the data definition by clicking the **Generate** icon .

Saving a data definition by adding basic generators with their modified settings

After you create an empty data definition in Test , you must add the basic generators with their modified settings, and save the data definition before you can generate test data.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an empty data definition that is opened for editing.

1. Drag a basic generator from the **Catalog** panel to the data definition.

   The basic generator is added as a field in the data definition.
2. Drag other basic generators from the **Catalog** panel to the data definition to complete the data structure that you want in the fabricated data.
3. Perform the following steps to modify the parameter configurations of a basic generator in a field of the data definition:

   a. Click a generator in the **Generator** column of a field where you want to customize the configuration.
   The parameters of the generator are displayed in the **Generator** dialog.

   b. Change the parameter settings in the **Generator** dialog to customize the generator.

   c. Close the **Generator** dialog.
   The changes in the generator configuration are automatically saved.
4. Click the **Save** icon to save the changes made to the Data definition or the **Save All** icon to save all changes made to different Data definitions by using the same Edit branch.

Note: When you edit different files that include datasets or Data definitions simultaneously, you can select your action from the following options:

To...	Do this...
Keep your current file open and close the open files on the right.	Right-click, and then click the **Close Tabs to right** option.
Keep your current file open and close the open files on the left.	Right-click, and then click the **Close Tabs to left** option.
Keep your current file open and close all the other open files.	Right-click, and then click the **Close Others** option.
Close all open file tabs.	Right-click, and then click the **Close All Tabs** option.

Note: The changes you made are only visible and available to you in the Edit branch. You must publish the changes in the Edit branch to the branch in the project repository for the changes to be visible to the other members of the project.

5. Click the **Publish** icon to publish the changes made to the data definition to the branch in the project repository.

Results

You edited and saved a data definition after adding basic generators and then modifying the configuration of the generators.

What to do next

You can perform any of the following tasks:

- Preview the test data generated by using the generators added as fields in the data definition by clicking the **Preview** icon .
- Generate the test data by using the generators added as fields in the data definition by clicking the **Generate** icon .

Modifying a data definition

After you create and publish a data definition in Test , you can modify the fields or the configuration of the generators in an existing data definition when you want to change the structure of the test data.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Click the data definition that you want to edit in the data navigator panel.

   Alternatively, right-click the data definition in the data navigator panel, and then click **Edit**.

   The selected data definition opens in the **Data** page.
5. Perform the actions that are listed in the following table to modify the data definition:

Modification	Action
Addition of new fields	Click the **Add field** option, and then enter a name for the field in the **Name** column. **Note:** A new row with an empty field is automatically displayed below the row that you edit.
Addition of generators in fields	Drag a generator from the **Catalog** panel into a field under the **Generator** column. **Note:** You can use the **Search catalog** field in the **Catalog** panel to find the required generator.
Replacement of generators in fields	Drag a generator from the **Catalog** panel into an existing generator field under the **Generator** column. The existing generator in the field is automatically replaced by the generator.
Modification of generator configurations	Click a generator in the **Generator** column. The generator configurations are displayed. Modify the options and settings in the generator configuration.

Modification	Action
	Note: If you modify the configurations of a basic generator, the changes are applicable only when the generator is used as field in a Data definition. The basic generator that is available in the catalog is not modified nor saved.
Inclusion of additional configuration details	Click the **Add new configuration** tab. Provide a name, and enter or select the other settings as an additional configuration to the generator.
Arrangement of fields	Drag and drop the field rows in the Data definition panel to rearrange the fields in the order you want.
Deletion of fields	Click the delete icon in the row of a field to remove the field from the Data definition.
Deletion of Data definition	Click the **Delete** icon in the Data definition panel to delete the Data definition from the existing location.

6. Click the **Save** icon to save the changes made to the Data definition or the **Save All** icon in the Data navigator panel to save all changes made to different Data definitions by using the same Edit branch.

Note: When editing multiple assets (such as datasets, data definitions or schemas) simultaneously, right-click a test asset in the Data page to manage open tabs using the following options:

To...	Do this...
Keep your current file open and close the open files on the right.	Right-click, and then click the **Close Tabs to right** option.
Keep your current file open and close the open files on the left.	Right-click, and then click the **Close Tabs to left** option.
Keep your current file open and close all the other open files.	Right-click, and then click the **Close Others** option.

To...	Do this...
Close all open file tabs.	Right-click, and then click the **Close All Tabs** option.

Note: The changes you made are only visible and available to you in the Edit branch. You must publish the changes in the Edit branch to the branch in the project repository for the changes to be visible to the other members of the project.

7. Click the **Publish** icon to publish the changes made to the data definition to the branch in the project repository.

Results

You modified an existing data definition in your project.

What to do next

You can generate test data by using the modified Data definition. See Generating fabricated test data on page 1012.

Viewing a data definition and its details

You can view the contents of a data definition on the Data page in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch from the **Branch** list.
   The navigation panel displays all the existing projects in the selected branch.
5. Find the data definition that you want to view, and then click it.

   You can either enter the name of the data definition in the **Search** field or expand the project and manually search in the data navigator panel.

The selected data definition opens in the **Data** page and the following elements are displayed:

Element	Description
The **Name** column	A list of fields and their custom names that exist in the Data definition is displayed in the column.
The **Generator** column	The generator that is selected for the corresponding field is displayed in the column. **Note:** If any of the fields does not have a corresponding generator, then you cannot preview or generate the test data for the Data definition.
The **Configuration details** column	The configuration details of each field are displayed in the column. **Note:** The column displays details only when additional configuration is added for the field. Also, if you have added three or more additional configurations, then the ellipses icon ... is displayed. You can hover over the icon to view the list of additional configuration names along with their distribution values. You can also click the column of a field to view the existing configuration details for the field in the corresponding *Generator* dialog.
Initials of members	The colored avatars, that represent the initials of the project members who are currently editing the data definition, are displayed in the page. You can hover over an avatar to view the name of the member and the branch that they are currently editing. This information can facilitate collaboration and communication among team members who are working on the same Data definition simultaneously. To understand how you can resolve conflicting edits, see Modifying a data definition on page 991.
The **Preview** icon	You can click the icon to preview the test data in the preview dialog of the **Data** page. The preview dialog contains the generate icon and the close icon. **Notes:**

Element	Description
	◦ If you are in an Edit branch, then you can access the **Generate** icon in the preview dialog only if the Data definition is saved. ◦ If you modify the Data definition but not save it, you can still preview the corresponding test data. However, you can generate the test data only after you save the Data definition.
The **Save** icon	You can click the icon to save the Data definition. If you modified the Data definition and you want to continue using the modified Data definition to generate test data without publishing it, you must save the changes. However, if you want to collaborate with other members of your project, you must publish the Data definition.
The **Generate** icon	You can click the icon to generate the test data in the CSV format. See Generating standard test data by using data definitions and exporting it to a .zip file on page 1014.
The **Delete** icon	You can click the icon to delete the Data definition. See Deleting a data definition on page 998.
The **Catalog** tab	Provides a structured overview of the available built-in and custom generators for the selected project.
The **History** tab	Provides an overview of all the previously generated test data. The following information about the history of the generated data is displayed: ◦ The month, date, and year of the generated test data. ◦ The status of the download that indicates whether the data generation was successful or failed. ◦ The number of rows in the generated test data. ◦ The initials of the project member who downloaded the generated test data. You can also use the following options to work with the past data: ◦ You can click the preview icon to review the already generated data before taking any further actions. ◦ You can click the generate icon to regenerate the same test data, if required. See Generating test data by using data definition history on page 1017. ◦ You can click the delete icon to delete specific test data history. You get a clutter-free view of the most relevant data and maintain your data generation history effectively.
The **Details** tab	Provides following details about the Data definition and its associated Git details:

Element	Description
	◦ Name of the branch that contains the data definition. ◦ Description for the Data definition, if any. ◦ Git-related details for the Data definition as follows: ▪ Asset ID: The ID uniquely identifies the data definition. You also have the option to copy the ID for easy reference or use. ▪ Repo: The repository name and path where the Data definition is stored is displayed. ▪ Path: The exact location of the data definition within the repository is displayed. ▪ Commit: The Git commit message, the name of the member who performed the changes, and the date and time of the commit are displayed. ▪ The copy button: You can click the button to copy the complete SHA commit ID for the data definition. ▪ The commit ID button: You can click the button to view the changes that were made in the commit directly within Git. ▪ The browse button: You can click the button to view the specific location in the Git history where the commit was performed for the data definition.
The Data definition tabs	When you open a Data definition, the name of the currently opened Data definition is displayed as a tab in the **Data** page. Beside this tab, additional tabs with the names of all other data definitions that are currently open on the **Data** page are also displayed. You can use these tabs to navigate between different Data definitions easily. Also, when you hover over the tab of a Data definition, a tooltip is displayed that provides the location and repository details where the Data definition is stored.
The **Branch** list	The list displays the name of the branch where you currently opened the Data definition. You can also use the list to choose and select a different branch from the available options, if required. You can switch between branches easily.

Note: The **Save** icon and the **Delete** icon are displayed in the Data definition page only if you are in an Edit branch. If you are in a branch other than the Edit branch, then an **Edit** icon is displayed in the Data definition. You can click the **Edit** icon to perform any modification to the Data definition. Also, if you do not have either the owner or the tester role for the Data definition, you cannot save, modify, or delete the Data definition.

Results

You have viewed the contents of the data definition in your project.

What to do next

You can perform any of the following tasks:

- Modify the data definition. See Modifying a data definition on page 991.

Deleting a data definition

You can delete a data definition when it is outdated or no longer relevant to your project. Additionally, when data definitions contain errors or inaccuracies that can lead to misleading or incorrect fabricated data, then you might want to remove them. This ensures that the project remains clutter-free, optimizes the performance of Test , and reduces storage requirements.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the data definition in the data navigator panel.
   Result
   The data dedfinition is displayed in the **Data** page
5. Click the **Delete** icon to delete the data definition.

   Alternatively, right-click the data definition in the data navigator panel, and then click **Delete**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 999.
6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

7. Click the **Delete** button in the **Delete Data definition** dialog.

Results

You have deleted a data definition from your project.

What to do next

You can publish the changes performed to the data definition (that is, deletion of the data definition) to the Git repository so that other members of the project do not see the deleted data definition. See Publishing all changes in an Edit branch on page 1023.

Management of schemas

A schema, also known as a database schema, defines how data is structured, organized, and managed in a database management system (DBMS). It outlines the format, storage, processing, security, and data access across various structural schema elements such as tables, views, and logical constraints.

The following sections provide an overview of how schema concepts are applied within Test Hub to support efficient data management and test design:

- Using schema in Test Hub on page 999
- Key benefits of schema in Test Hub on page 1000
- Primary key on page 1000
- Foreign key on page 1000

Using schema in Test Hub

- You can create a new schema either as an empty structure or by connecting it to an existing database, allowing you to visualize how data is structured and related in real environments. See Creating an empty schema on page 1000 and Creating a schema by using a database connection on page 1002.
- After the schema is created, two separate tabs are displayed based on the database connection:
  - **Database View**
  - **Schema View**
- You can add data definitions and establish relationships between their fields in a schema.

Key benefits of schema in Test Hub

The schema in Test offers the following advantages:

- It defines how data is stored and related within the database, making it easier to understand, manage, and maintain.
- You can group similar data to reuse it across different scenarios, even when it serves different purposes.
- You can enhance data integrity by using logical constraints such as tables, primary keys, and relationships that helps maintain data consistency and accuracy while reducing errors.

To use schema in Test, you must be aware of the terms and concepts that are used to link database tables.

Primary key

A primary key is one or more columns in a database table that uniquely identify each record in that table. The primary key enforces data integrity by ensuring that no two rows have the same value(s) in the key column(s) and that these values are never null. Database systems use the primary key to efficiently retrieve, update, or relate records across multiple tables in a relational database.

Foreign key

A foreign key is a link between two tables that ensures the data in one table corresponds with the data in another table, thereby maintaining uniformity of the database information.

Creating an empty schema

You can create an empty schema to define a structured data storage. Initially, the schema is not connected to any database.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.

4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.

5. Perform any of the following actions:
   - Click **Add files** and select **Schema** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Schema**.
   - Right-click the application in the data navigator panel, go to **New**, and select **Schema**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 1001.

6. Select your action to either select an existing Edit branch or create an Edit branch:

   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

        Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

     c. Click **Ok**.
   - Perform the following steps to create the Edit branch:
     a. Select **Create new edit branch**.
     b. Enter a name for the Edit branch in the **Branch Name** field.
     c. Click **Save**.

   Result

   The **Create a Schema** dialog is displayed.

7. In **Step 1** of the **Create a Schema** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a schema.
   b. Select **Location** from the drop-down list to store the schema.
   c. Type a unique **Name** for the schema.
   d. Type a brief description of the schema in the **Description** field.
   e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create a Schema** dialog, perform the following tasks:
   a. Select **Source** as **Create an Empty Schema**.
   b. Click **Create**.

Results

You have created an empty schema.

What to do next

You can add a database connection to an empty schema. See Adding a database connection to the schema on page 1002.

Adding a database connection to the schema

You can add a database connection to link a schema to a database for data generation and management.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Added a database connection in your project. See Adding a database connection on page 938.
- Created an empty schema in your project. See Creating an empty schema on page 1000.

About this task

When you create an empty schema, it is not linked to any database. In the **Database View** tab, a message is displayed that indicates that the schema is empty and requires a database connection.

1. Select the **Schema** in the data navigator panel that is not connected to any database.
   Result
   The selected schema is displayed on the **Data** page.
2. Navigate to the **Database View** tab.
   If the schema is not connected to any database, a message appears `This schema is not linked to any database.`
3. Click **Add DB Connection**.
   The **Add a Database Connection** dialog is displayed.
4. Select a database from the **Select a Database Connection** list.
5. Click **Done**.

Results

You have added a database connection to the schema.

Related information

Creating a schema by using a database connection on page 1002

Creating a schema by using a database connection

When your tests require data that reflects existing database models, you can create a schema by connecting to an existing database. You can retrieve and manage live data structures that help your test data reflect production schema complexity and relationships more accurately.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Read information about using the schema. See Management of schemas on page 999.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created an application in your project. See Creating an application on page 766.
- Added a database connection in your project. See Adding a database connection on page 938.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch in the project repository from the **Branch** list.

   The assets and resources in the selected branch are displayed in the data navigator panel.
5. Perform any of the following actions:
   - Click **Add files** and select **Schema** on the **Test Editor** page.
   - Click the **Action Menu** icon and select **Schema**.
   - Right-click the application in the data navigator panel, select **New > Schema**.

   The **Edit Branch** dialog is displayed.

   If the **Edit branch** dialog is not displayed, then go to step 7 on page 1004.
6. Select your action to either select an existing Edit branch or create an Edit branch:
   - Perform the following steps to select an existing Edit branch:
     a. Select **Use an existing edit branch**.
     b. Select an Edit branch from the **Select branch** list if Edit branches already exist for your repository.

Note: A warning message is displayed if the Edit branch, which you selected, is behind the source branch. To update the selected edit branch and maintain synchronization with the source branch, you can select the **Update branch option when switching** checkbox.

c. Click **Ok**.

- Perform the following steps to create the Edit branch:
  a. Select **Create new edit branch**.
  b. Enter a name for the Edit branch in the **Branch Name** field.
  c. Click **Save**.

Result

The **Create a Schema** dialog is displayed.

7. In **Step 1** of the **Create a Schema** dialog, perform the following tasks:
   a. Select the **Application** for which you want to create a schema.
   b. Select **Location** from the drop-down list to store the schema.
   c. Type a unique **Name** for the schema.
   d. Type a brief description of the schema in the **Description** field.
   e. Click **Next** to proceed to Step 2.
8. In **Step 2** of the **Create a Schema** dialog, perform the following tasks:
   a. Select **Source** as **Select a Database Connection**.
   b. Select a database from the **Select a Database Connection** list.
   c. Click **Create**.

Results

You have created a schema by using an existing database connection.

Related information

Managing data definitions in a schema on page 1004

Managing data definitions in a schema

You can manage and organize data definitions within a schema to support consistent and accurate test data generation.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that your project contains a schema.

About this task

You can access a schema in your project and perform actions such as creating or importing data definitions, linking fields, modifying generator settings, and saving schema changes. You can design data relationships by using the **Schema View**, or use the **Database View** to align data definitions with your database structure.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.
   The **Data** page and the data navigator panel are displayed.
4. Select the schema that you want to edit in the data navigator panel.

   The selected schema opens in the **Data** page.
5. You can perform the actions that are listed in the following table to modify data definitions in a **Schema View**:

Task	**Action**
Create an inline data definition	Perform the following steps: a. Click **Add Data Definition**. b. Type a name in the **Name** field under the **Create Inline Data Definition** section. c. Click **Add**. A table with an empty field is displayed in the schema view.
Import existing data definitions	Perform the following steps: a. Click the **Add Data Definition**. b. Select the data defintions from the **Available Data Definitions** section. c. Click **Add**. **Note:** You can use the **Search** field in the **Add Data Definition** dialog to find the required data definition.
Add a relationship between fields	Perform the following steps: a. Hover over the field line until the **Drag to connect to a field** tooltip appears. b. Drag to the destination field in another data definition

Task	Action
	Note: The data type of the source field and the destination field must be the same.
Define the relationship ratio	Perform the following steps: a. Click the **Destination** field. b. Enter the distribution ratio (For example: One-to-One or One-to-Many) in the **Relationship configuration** dialog.
Convert between Referenced and Inline DDFs	Perform the following steps: a. Click the **Actions-menu** icon on the data definition. ▪ Select **Convert to Referenced**, if the data definition is **inline** ▪ Select **Convert to Inline**, if the data definition is referenced.
Rename inline data definition	Perform the following steps: a. Click the **Actions-menu** icon on the data definition. b. Select **Rename**. c. Type a new name in the **Rename Data Definition** dialog. d. Click **Rename**.
Add new fields	Perform the following steps: a. Click the **Add field** icon . b. Type a field name in the **Name** column. An empty row is automatically inserted below the current row for your next entry.
Add generators in fields	Drag a generator from the **Catalog** panel into a field under the **Generator** column. **Note:** You can use the **Search Catalog** field in the **Catalog** panel to find the required generator.
Replace generators in fields	Drag a generator from the **Catalog** panel into an existing generator field under the **Generator** column. The existing generator in the field is automatically replaced by the generator.
Modify generator configurations	Perform the following steps:

Task	Action
	a. Click a generator in the **Generator** column. b. Update the configuration settings in the panel. **Note:** If you modify the configurations of a basic generator, the changes are applicable only when the generator is used as a field in a data definition. The basic generator that is available in the catalog is not modified or saved.
View sample data for a generator in Schema View	Perform the following steps: a. Click the generator label next to a field in the table in **Schema View**. The properties of the generator is displayed. b. Click the **View sample data**. A list of five sample records is displayed, fetched from the source database. c. **Optional**: Click the **Reload Sample Data** icon to refresh the data from the source.
Include additional configuration details	Perform the following steps: a. Click the **Add new configuration** tab. b. Type a name and set other settings as an additional configuration to the generator.
Reorder fields	Drag and drop rows in the table to rearrange the fields.
Delete fields	Click the **Delete** icon in the field row to remove the field.
Remove one or more data definitions from Schema View	Perform the following steps: a. Click the **Actions-menu** icon on the data definition. b. Select **Remove from schema**. The data definition is removed from the schema. Alternatively, select the data definition and click **Remove (1)**. c. Select the data definitions or that you want to remove and click **Remove (#)**. The number in parentheses indicates the number of data definitions that are currently selected. For example, **Remove (2)** or **Remove (5)**. To remove all the data definitions from the Schema View, perform the following steps:

Task	Action
	a. Select the data definitions in the Schema View by pressing **Ctrl+A** to select all, or by pressing **Ctrl** and clicking on each DDF that you want to select. b. Click **Remove All** to remove all the data definition from the schema. The **Remove All Data Definitions** dialog box is displayed. c. Click **Yes** to confirm. **Note:** You cannot undo this action after the data definitions are removed.
Update data definitions in Schema View	Perform the following steps: a. Click the **Update** icon to update the default row counts and default distribution ratios for the data definitions within the schema. The **Update Schema** dialog box is displayed, with the **Default row counts** and **Default distribution ratios** selected. b. Click **Update**.

6. You can perform the actions that are listed in the following table to modify data definitions in a **Database View**:

Task	Action
Export a data definition to a schema	Perform the following steps: a. Click the **Actions-menu** icon on the data definition. b. Select **Add table to schema**. **Note:** The exported data definition changes to green color.
Select the linked data definitions	Perform the following steps: a. Click the **Actions-menu** icon on the data definition. b. Select **Select linked tables**. **Note:** The linked data definitions are changed to blue color.
Add missing columns	Perform the following steps: a. Select the data definition exported to schema.

Task	Action
	b. Click the **Actions-menu** icon on the data definition. c. Select **Add missing columns to schema**.
Remove unnecessary fields	Perform the following steps: a. Select the data definition exported to schema. b. Click the **Actions-menu** icon on the data definition. c. Select **Remove unnecessary fields**.

7. Click the **Save** icon to save the changes made to the schema.

Results

You have updated the data definition within a schema.

Related information

Publishing schema changes

Auto-assigning generators and AI-provided data configurations to data definitions in the Schema View

You can use AI assistance to automatically assign generators and AI-provided data configurations to the fields in data definitions within the Schema View.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.
- Ensured that your project contains a Web UI test or suite.
- Ensured that the required variables have been created and are included in your Web UI test or suite.
- Ensured that your project contains a dataset.
- Ensured that you have configured the required AI provider credentials. For more information, see Adding cloud credentials of AI providers. on page 1122

About this task

When a schema contains multiple data definitions and fields, manually assigning generators and configuring them can be time-consuming. AI-based inference analyzes the available field information and recommends suitable generators and data configurations for the selected data definitions.

Note: Existing generators remain unchanged unless you select the option to overwrite them.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.
   The **Data** page and the data navigator panel are displayed.
4. Select the schema that you want to edit in the data navigator panel.

   The selected schema opens in the **Data** page.
5. Click the **Update** icon .
6. Select the **Assign generators to fields** checkbox.

   The **AI Provider** option is displayed.
7. Select the configured **AI Provider** from the drop-down list.

   Note: When you select an **AI provider**, sample data from the selected tables is shared with the AI service to configure generators.
8. **Optional:** Select the **Overwrite existing generators** checkbox to overwrite the existing generators.
9. Click **Update**.

   Note: Verify the AI-provided generator configurations to ensure their accuracy, as AI-produced data might contain errors.

Results

You have automatically assigned generators and AI-provided data configurations to the fields in the selected data definitions.

Deleting a schema

You can delete a schema when it is outdated or no longer relevant to your project. Additionally, if a schema contains errors or inaccuracies that might lead to the generation of misleading or incorrect fabricated data, then you might want to remove them. By removing such shemas, you can maintain a clean and organized workspace, optimize the performance of Test , and reduce unnecessary storage usage.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735
- Ensured that your project contains a schema.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the schema that you want to delete from the data navigator panel.

   The selected schema opens on the **Data** page.
5. Click the **Delete** icon .

   Alternatively, right-click the schema in the data navigator panel, and then click **Delete**.

   The **Delete Schema** dialog is displayed.
6. Click **Delete**.

Results

You have deleted a schema from your project repository.

Publishing changes in data definitions, datasets, or schemas

After creating or modifying a data definition, dataset, or schema in Test , you can publish changes from an Edit branch to the source branch in the project repository. After you publish the changes, other members of the project can view and work with the updated datasets.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created or modified a schema, dataset, or data definition.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.

3. Go to **Author > Data**.

   Result

   The **Data** page and data navigator panel are displayed.

4. Select the data asset (schema, data definition, or dataset) you want to publish from the data navigator panel.

   Result

   The selected data asset is displayed on the **Data** page.

5. Click the **Save** icon to save changes.

6. Click the **Publish** icon .

   The **Publish** dialog is displayed.

7. Type the description in the **Description of change** field.

8. Click **Publish**.

   Alternatively, you can publish the changes from the **Changes** tab.

   If the publish succeeds, the Edit branch is automatically deleted and removed from the **Branch** list and the changes are merged into the previously selected remote branch.

   If the publish fails due to a conflict with the remote branch, click **Push** in the **Publish** dialog to push the edit branch to the remote repository. Then, you can create a pull request and resolve the conflicts manually in Git before you merge the changes into the target branch.

Results

You have published the changes in an Edit branch to the remote branch in the project repository.

Related information

Publishing all changes in an Edit branch on page 1023

Generating fabricated test data

You can generate fabricated test data in Test to support a wide range of test scenarios. The generated data is saved as a CSV file on your system and can be used to override existing datasets during test execution.

Data fabrication is the process of creating synthetic test data that mimics real-world information without exposing actual user or production data. You can simulate various testing scenarios using structured, configurable, and reusable datasets. In Test , this process involves defining data structures, applying data generation logic through built-in or custom generators, and producing test-ready datasets that enhance both test coverage and data privacy.

By combining datasets, data definitions, database query and schemas, Test Hub helps you to create realistic, privacy-compliant, and reusable test datasets that improve test coverage and accuracy.

You can generate fabricated test data in the following ways:

- Generating test data by using data definitions and exporting it to a `.zip` file.
- Generation of test data using Data definition history.
- Generation of schema based test data and exporting it as a `.zip` file.
- Generating test data by using data definitions or schema and exporting it to a database.

Generating test data by using data definitions and exporting it to a `.zip` file

You can create a data definition, add fields, and configure generators to produce realistic test data. Before generating, you can preview the data to ensure it meets your expectations. If the fields have no generators, you can still save the definition, but preview and generation are only available after at least one generator is configured.

Note: While configuring the generator parameters for a field in a Data definition, you must use the **Allowed** option of the **Duplicates** field of the generator with caution, particularly when you configure the data generator with a narrow criteria for data generation. If you enable the **Allowed** option in scenarios where duplications are not feasible within the defined criteria, then the preview or generation of test data might not be successful.

Also, the data distribution in a generated test data closely aligns with the configured weightage that you provide in the **Distribution** field of the generator. Therefore, the accuracy of the test data improves as more number of records are generated. See Basic generators on page 874.

See Generating standard test data by using data definitions and exporting it to a .zip file on page 1014.

Generation of test data using Data definition history

You can use the **History** tab in Test to view all previously generated test data for a Data definition. From this tab, you can select and regenerate any earlier version of test data without reconfiguring the Data definition. This helps ensure consistency during repeated test runs, simplifies troubleshooting, and supports better collaboration when multiple team members need access to the same test dataset.

See Generating test data by using data definition history on page 1017.

Generation of schema based test data and exporting it as a `.zip` file

You can generate test data based on a schema in Test and export it as a `.zip` file containing CSV datasets. This allows you to use the fabricated data outside Test Hub or share it across different environments.

See Generating and exporting schema-based test data to a .zip file on page 1019.

Generating test data by using data definitions or schema and exporting it to a database

You can generate test data from a schema in Test and export it directly to a connected database. This enables seamless population of test environments with realistic, structured data for end-to-end testing.

See Generating test data by using data definitions or schema and exporting it to a database on page 1020.

Generating standard test data by using data definitions and exporting it to a `.zip` file

You can generate synthetic standard test data by using data definitions in Test to simulate realistic test scenarios, preview data, and export it for use in your testing workflows.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch from the **Branch** list.
   The navigation panel displays all the existing projects in the selected branch.
5. Find the data definition that you want to view, and then click it.

   Note: You can either enter the name of the data definition in the **Search** field or expand the project and manually search in the Data navigator panel.

   Result
   The selected data definition opens in the **Data** page.
6. Click the **Preview** icon to preview the test data that you want to generate.

   The test data is generated in the preview dialog.

   Note: You can preview a test data to ensure whether the data is in the expected format. You can have a glimpse of the generated test data that can help you to identify any potential issues, inconsistencies, or errors in the Data definitions. If you modify the data definition but not save it, you can still preview the corresponding test data. However, you can generate the test data only after you save the data definition.

7. Click the **Standard** tab in the preview section to view the generated standard test data.
8. Click the **Generate** icon either from the **Data** page or from the preview page.
   The **Generate data** dialog is displayed.

Note: If you are in an Edit branch, then you can access the **Generate** icon from the preview dialog only if the data definition is saved.

9. Perform the following actions in the **Generate data** dialog:
    a. Enter a name in the **Name** field to identify the test data when generated.
    b. Enter the number of rows that you want to generate in the **Number of Rows** field.
    c. **Optional**: Enter the seed value in the **Seed** field.

    Whenever you generate test data to perform application testing, the generated data is different. You can produce the same set of data multiple times by setting the seed value.

    For example, assume that you have used the seed value as `1` to generate test data from a data definition. When you want to create another test data with the same set of values, then you can provide the seed value as `1` during the creation of the test data.

    d. Clear the **Include headers** option if you do not want to include headers in the generated test data.
    e. Click the **Download** option to initiate the generation of the test data.

Results

The generated test data is downloaded into your local file system as a Comma Separated Values (CSV) file.

Note: You can use other applications that support viewing or editing of the CSV files. You must ensure to apply the relevant formats to the fields to view the correct format of the generated test data.

What to do next

You can view the information about the history of the generated data in the **History** tab of the **Data** page.

Generating negative test data for a data definition

You can generate synthetic negative test data for a data definition in Test to simulate invalid and boundary test scenarios for application forms.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

The **Overview** page of the project is displayed.

3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.

4. Select the branch from the **Branch** list.
   The navigation panel displays all the existing projects in the selected branch.
5. Find the data definition that you want to view, and then click it.

   Note: You can either enter the name of the data definition in the **Search** field or expand the project and manually search in the Data navigator panel.

   Result
   The selected data definition opens in the **Data** page.
6. Click the **Preview** icon to preview the test data that you want to generate.

   The negative test data is generated in the preview dialog.

   Note: You can preview test data to ensure that the data is in the expected format. You can have a glimpse of the generated test data that can help you to identify any potential issues, inconsistencies, or errors in the Data definitions. If you modify the data definition and do not save it, you can still preview the corresponding test data. However, you can generate the test data only after you save the data definition.

7. Click the **Negative** tab in the preview section to view the generated negative test data.
8. Click the **Generate** icon either from the **Data** page or from the preview page.
   The **Generate data** dialog is displayed.

   Note: If you are in an Edit branch, then you can access the **Generate** icon from the preview dialog only if the data definition is saved.

9. Perform the following actions in the **Generate Data** dialog:
   a. Enter a name in the **Name** field to identify the test data when generated.
   b. Enter the number of rows that you want to generate in the **Number of Rows** field.
   c. **Optional**: Enter the seed value in the **Seed** field.

      Whenever you generate test data to perform application testing, the generated data is different. You can produce the same set of data multiple times by setting the seed value.

      For example, assume that you have used the seed value as `1` to generate test data from a data definition. When you want to create another test data with the same set of values, then you can provide the seed value as `1` during the creation of the test data.

d. **Optional**: Select the following options as required:
   - **Include Headers** - Includes column headers in the generated output.
   - **One per Row** - Generates one invalid value per row while keeping the remaining field values valid.
   - **Include Metadata** - Includes metadata information in the generated output, such as the column that contains the negative data and the property used to generate the negative data.

e. Click the **Download** option to initiate the generation of the test data.

Results

The generated test data is downloaded into your local file system as a Comma-Separated Values (CSV) file.

Note: You can use other applications that support viewing or editing of the CSV files. You must ensure to apply the relevant formats to the fields to view the correct format of the generated test data.

What to do next

You can view the information about the history of the generated data in the **History** tab of the **Data** page.

Generating test data by using data definition history

You can generate fabricated test data in Test by using the history of existing data definitions. This approach supports better collaboration by allowing multiple team members to work with consistent test data at different stages of the project.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a data definition. See Creating an empty data definition on page 980.
- Generated test data by using a data definition. See Generating standard test data by using data definitions and exporting it to a .zip file on page 1014.

Note: If a project member has generated a test data from a data definition, then the other project members can view the generated test data only when the changes are pushed to the repository.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch from the **Branch** list.

The navigation panel displays all the existing projects in the selected branch.

5. Find the data definition that you want to use for regenerating test data, and then click it.

   Note: You can either enter the name of the data definition in the **Search** field or expand the project and manually search in the data navigator panel.

   Result

   The selected data definition opens in the **Data** page.

6. Click the **History** tab.

   A list of all the previously generated test data are displayed, along with the following information:

   - The month, date, and year of the generated test data.
   - The status of the download action that indicates whether the data generation was successful or failed.
   - The number of rows in the generated test data.
   - The initials of the project member who downloaded the generated test data.

7. Perform the following actions in the **History** tab:

   a. Find the required test data.

   b. **Optional**: Click the **Preview** icon to preview the test data before regeneration.

      The test data is generated in the preview dialog.

      Note: You can preview the previosuly generated test data to ensure whether the data is in the expected format. You can have a glimpse of the regenerated test data that can help you to identify any potential issues, inconsistencies, or errors in the data definitions.

   c. Click the **Generate** icon .

      The **Generate data** dialog is displayed.

      Note: If you are in an Edit branch, then you can access the **Generate** icon from the preview dialog only if the data definition is saved.

8. Perform the following actions in the **Generate data** dialog:

   a. Enter a name in the **Name** field to identify the test data when generated.

   b. Enter the number of rows that you want to generate in the **Number of Rows** field.

   c. **Optional**: Enter the seed value in the **Seed** field.

      Whenever you generate test data to perform application testing, the generated data is different. You can produce the same set of data multiple times by setting the seed value.

For example, assume that you have used the seed value as `1` to generate test data from a data definition. When you want to create another test data with the same set of values, then you can provide the seed value as `1` during the creation of the test data.

d. Clear the **Include headers** option if you do not want to include headers in the generated test data.
e. Click the **Download** option to initiate the generation of the test data.

Results

The generated test data is downloaded into your local file system as a Comma Separated Values (CSV) file.

Note: You can use other applications that support viewing or editing of the CSV files. You must ensure to apply the relevant formats to the fields to view the correct format of the generated test data.

Generating and exporting schema-based test data to a `.zip` file

You can generate schema-based test data and export it as a `.zip` file that contains `.csv` files. The exported data remains consistent with the schema structure, which supports accurate testing, data validation, and you can use this data to perform data-driven testing across multiple scenarios in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that your project contains a schema.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch from the **Branch** list.
   The navigation panel displays all the existing projects in the selected branch.
5. Select the schema from which you want to generate and export test data in the data navigator panel.

   The selected schema is displayed in the **Data** page.

Note: You can either enter the name of the Schema in the **Search** field or expand the project and manually search in the data navigator panel.

6. Click the **Preview** icon to preview the test data that you want to generate.

   The test data is generated in the preview dialog.

   Note: You can preview test data to ensure whether the data is in the expected format. You can review the generated test data that can help you to identify any potential issues, inconsistencies, or errors in the schema. If you modify the schema and do not save it, you can still preview the corresponding test data. However, you can generate the test data only after you save the schema.

7. Click the **Generate** icon from the **Data** page.
   The **Generate data** dialog is displayed.
8. Select **Download as a zip file** from the **Select destination** list.
9. Type the file name in **File Name** field.
10. **Optional**: Enter the seed value in the **Seed** field.
    For example, consider that you have used the seed value of `1` to generate data from a schema. When you want to create another data with the same set of values, then you can provide the seed value as `1` during the creation of the test data.
11. Enter the number of rows that you want to generate in the **Generate** column.
12. **Optional**: Enter a value in the **Target** field to customize the number of rows to be generated in the tables.
    The default value in the **Target** field is set to 100%.

    For example, if a table in the database contains 10 rows and if you set the **Target** field to 200%, then the generated data displays 20 rows.

13. Click **Download** to initiate the generation of the data.
    Result
    The generated data is downloaded into your local file system as a `.zip` file.

Results

You have generated realistic and schema-aligned test data and exported it as a `.zip` file that contains `.csv` files.

What to do next

You can create a dataset by using the downloaded `.csv` files. See Creating a dataset by using a CSV file on page 956.

Generating test data by using data definitions or schema and exporting it to a database

You can generate test data by using data defintions or and schema, and export it directly to a connected database in Test . This process supports populating or updating database tables with realistic and schema-aligned test data used in development, testing, or validation workflows.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that your project contains data definition or schema.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Author > Data**.

   The **Data** page and the data navigator panel are displayed.
4. Select the branch from the **Branch** list.
   The navigation panel displays all the existing projects in the selected branch.
5. Select the data definition or schema from which you want to generate and export test data in the data navigator panel.

   The selected data definition or schema is displayed in the **Data** page.

   Note: You can either enter the name of the data definition or schema in the **Search** field or expand the project and manually search in the data navigator panel.

6. Click the **Preview** icon to preview the test data that you want to generate.

   The test data is generated in the preview dialog.

   Note: You can preview test data to ensure whether the data is in the expected format. If you modify the data definition or schema and do not save it, you can still preview the corresponding test data. However, you can generate the test data only after you save.

7. Click the **Generate** icon on the **Data** page.
   The **Generate data** dialog is displayed.
8. Perform the following steps to generate test data by using data definitions:
   a. Select the appropriate database from the **Select a Database Connection** list.

   b. Select a table from the **Table** dropdown.

      The selected table name appears in the **Table** field.

If the selected table structure does not match the data definition for data generation, a **Mismatch** indicator is displayed next to the table name, and the **Export** button is disabled.

The **Current Contents** field displays the existing number of rows in the selected table.

c. Enter the number of rows you want to generate in the **Number of Rows** field.

9. Perform the following steps to generate schema based test data:
   a. Select the appropriate database from the **Select a Database Connection** list.

      The **Tables** column lists the available tables from the schema.

      The **Current** column displays the number of existing rows in each table.

   b. Enter the number of rows to generate for each table in the **Generate** column.

      If there is a mismatch between the selected table and the schema used for data generation, a **Schema mismatch** indicator is dispalyed, and you cannot generate data for that table.

   c. Enter a value in the **Target** field to customize the number of rows to be generated in the tables.

      The default value in the Target field is set to 100%.

      For example, if a table in the database contains 10 rows and if you set the Target field to 200%, then the generated data displays 20 rows.

10. **Optional**: Enter the seed value in the **Seed** field.
    Whenever you generate data to perform application testing, the generated data is different. You can produce the same set of data multiple times by setting the seed value.
    For example, consider that you have used the seed value of `1` to generate data from a data defintion or schema. When you want to create another data with the same set of values, then you can provide the seed value as `1` during the creation of the test data.
11. **Optional**: Select the **Overwrite** checkbox to overwrite the rows in the tables for generating data.
    This option is not selected initially.
    For example, if a table in the database contains 10 rows and if you select the **Overwrite** checkbox, then the generated data replaces the existing 10 rows with new data generated.
12. Click the **Export** option to initiate the generation of the test data.
    The generated data is exported to the database.

Results

You have generated the test data by using data definition or schema and exported it to directly to the connected database.

What to do next

You can create a dataset by using the database query. See Creating a dataset by using a database query on page 960.

Publishing all changes in an Edit branch

After you create or modify any data or test assets in an Edit branch of a project in Test , you can publish all the changes that are in an Edit branch to the source branch in the project repository. After publishing, other members in the project can view and use the resources.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created or modified a test or data asset.
- Read and understood the information in Edit branch overview on page 760 and Create or select an Edit branch on page 772.

About this task

When you create or modify multiple assets in a project, the **Changes** page lists all the changed assets in the selected branch. In this scenario, you cannot publish or discard a single change. You must either publish or discard all changes that are made to all the listed assets.

When you review assets on the **Changes** page before publishing, each executable asset displays a colored status indicator in the parent row as follows:

- **Green** – The most recent run passed.
- **Red** – The most recent run failed.
- **Orange** – The most recent run resulted in an error.
- **Grey** – No run result is available for this revision.

The indicator helps you view the most recent run result for that specific asset revision. The colored bar on the left side of the parent row represents the most recent run result.

When you review assets on the **Changes** page before publishing, each executable asset displays a coloured status indicator in the parent row. The indicator helps you understand the most recent run result for that specific revision of the asset. The coloured bar on the left side of the parent row represents the latest run result:

- **Green** – The most recent run passed.
- **Red** – The most recent run failed.
- **Orange** – The most recent run resulted in an error.
- **Grey** – No run result is available for this revision.

The **Changes** page also provides you with information about the number of commits that the selected branch is ahead and behind. After you publish an asset to the Git repository, all the other project members who have access to that asset can view the updated asset. You can perform the following actions in the **Changes** page:

Options	Actions
Edit	Click **Edit** to make any updates to the asset before publishing.
Discard	Click **Discard** to remove the changes that you made to the asset.
Save as	Click **Save as** to save a copy of the asset. **Note:** When you try to publish a modified asset with conflicting edits, an error about conflicting changes is displayed. However, you can use the **Save as** option to save a copy of the asset edits under a new asset name. You must discard the edits that you made to the original asset. For example, consider a scenario when Amy Smith and John Doe edit the same Data definition, and Amy Smith edits and publishes the Data definition. On the **Changes** page, a message that indicates the number of commits ahead and behind in the branch is displayed when John Doe tries to publish the same Data definition. You can then use the **Save as** option to resolve this issue.
Compare	Click **Compare** to view the differences between the asset on the edit branch and the asset on the main branch.
Update	Click **Update** to merge the changes from remote branch that are not in your Edit branch.
Details	Click **Details** to see the changes.
Reset	Click **Reset** to discard the changes
Push	Click **Push** to upload the edit branch to remote repository when a conflict prevents publishing.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Go to **Author > Changes**.
4. Select the Edit branch from the **Branch** list.
   All the updated assets that are present in the selected branch are displayed in the **Changes** page.
5. Click **Publish**.
   The **Publish** dialog is displayed.

6. Enter a description for the changes that you performed to the asset, and then click **Publish**.
   If the publish succeeds, the Edit branch is automatically deleted and removed from the **Branch** list and the changes are merged into the previously selected remote branch.
   If the publish fails due to a conflict with the remote branch, click **Push** in the **Publish** dialog to push the edit branch to the remote repository. Then, you can create a pull request and resolve the conflicts manually in Git before you merge the changes into the target branch.

Results

You have published all the changes in an Edit branch to the remote branch in the project repository.

Test Execution Specialist Guide

This guide describes tasks that you must complete before you can configure and run tests in Test . You can find information about configuring runs for the different test types that are supported. You can also find information about other tasks that you can perform on the Resource Monitoring Service, Virtualization, and Integrations with third-party applications. This guide is intended for testers or test execution specialists.

System modeling

You can use the system model feature on Test to create a model that simulates an application or system under test. You can define components to represent the actual components in the system under test, define their relationships and then view the components in a system under test as a diagrammatic representation.

Introduction to a system model

A system model is a logical presentation of the components contained in a system under test and the relationships between components. You can create a system model when you want to visualize the different components and their relationships as a diagrammatic representation.

You can create only one system model within a team space on Test. A team space repository is required to contain the system model that you create. Before you can create a system model, you must add a team space repository. The system model that is created or edited is stored in the team space repository. All projects within a team space share the same system model. All members of a team space can create, edit, and view the system model.

After you log in to Test, you can go to the **System Model** page to create a system model for your team space. Alternatively, you can go to any of the projects in your team space and create a system model from the **System Model** page within a project. The system model is accessible from all the projects in your team space.

Components in a system model

Components are the basic building blocks that are used to represent a piece of software in an application.

You can model a system by using components. You can create components that represent the type of asset or resource used by the system or application under test and can be of the following types:

Component type	Description
Database	The component type represents the database resource. You can select this component type when you want to represent a database asset or resource in the system model.
Service	The component type represents the virtual service resource. You can select this component type in the system model to represent a virtual service in the team space repository or associate a virtual service resource that is in the project repository.
UI	The component type represents the user interface (UI) resource. You can select this component type when you want to represent the UI resource in the system model.

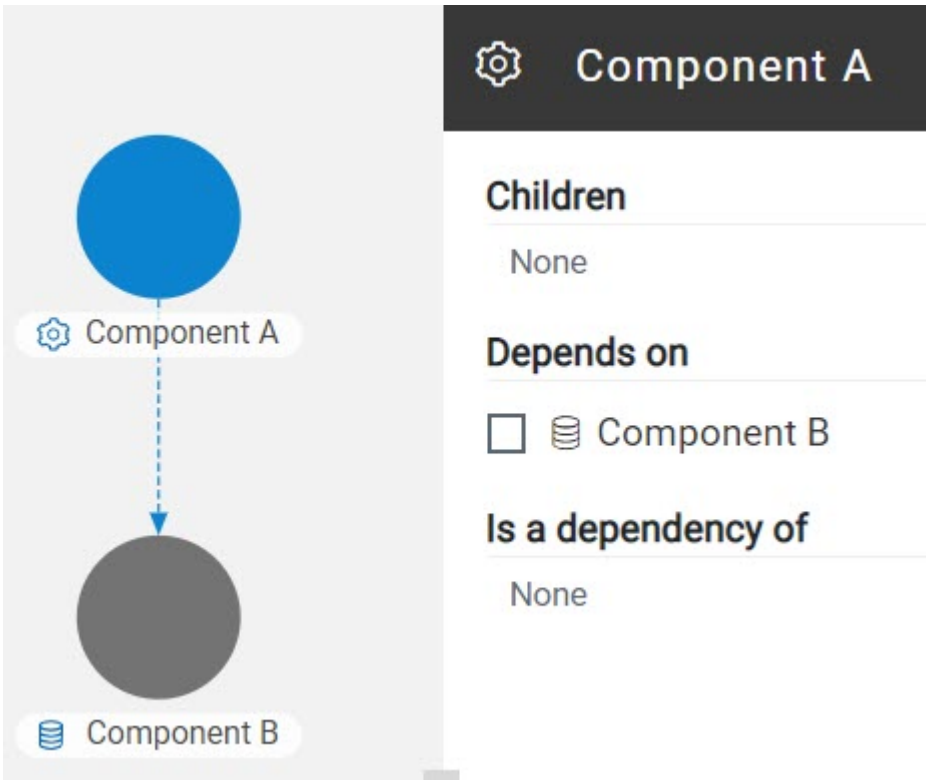
Component relationships

You can create multiple components to depict the different resources in the system or application under test. You can create a component as a *child* component of a *parent* component. You can also create multiple components as children components of the same parent component.

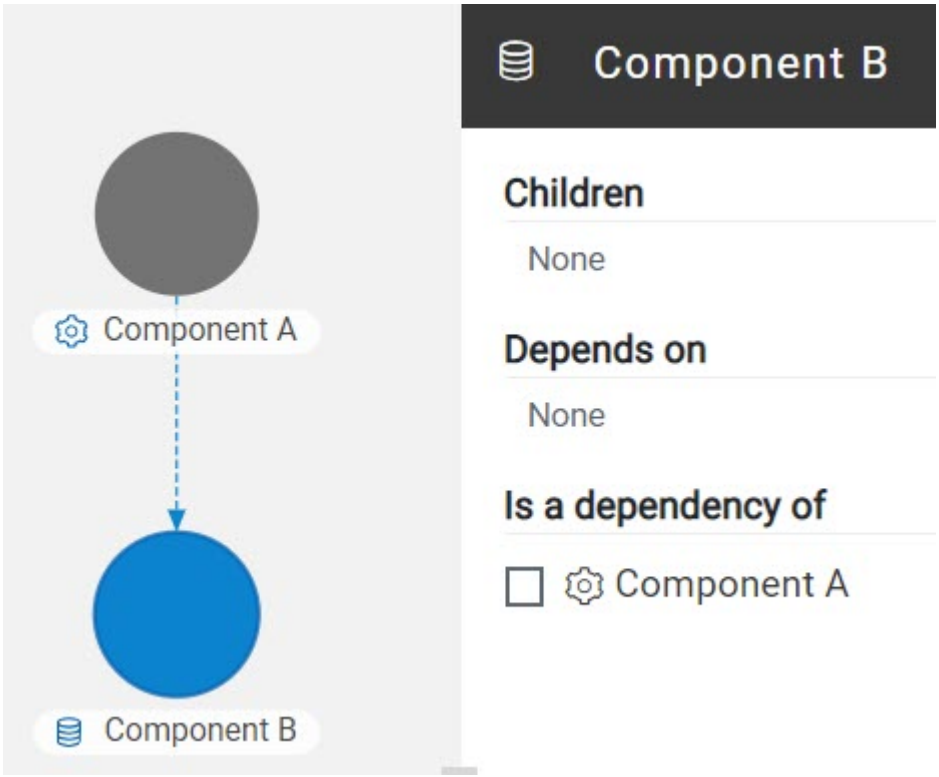
You can define the relationship between components if one component depends on another or if a component has a dependency of another component. The **Depends on** or **Is a dependency of** relationship depends on the component that you select in the system model after you define the relation.

For example, if you define *Component A* to depends on *Component B*, then the relationship is displayed with a line arrow from *Component A* to *Component B*, and the arrow points to *Component B*.

When you click the *Component A* in the system model page, you can view that the *Component B* is listed under the **Depends on** section for *Component A* in the right pane.



When you click the *Component B* in the system model, you can view that the *Component A* is listed under the **Is a dependency of** section for *Component B* in the right pane.



Component associations

When you work with the system model from your project in a team space, you can associate test resources that are in your project repository with components in the system model. You can associate all types of test resources such as Suites or tests, and virtual service resources with the components.

After associating test resources, you can view them when you select the component with which they are associated. You can select the associated test resources to start a run of the Suites or tests or start an instance of the virtual services resource.

Using the system model UI

The following table lists the icons used in a system model and the icons that are displayed on the **System Model** page:

Icon	Description
	Represents a system model.
	Represents Suites and tests in a repository.
	Represents a database resource. The component name is prefixed with this icon when you select the component type as database.
	Represents a service resource. The component name is prefixed with this icon when you select the component type as service.
	Represents a user interface (UI) resource. The component name is prefixed with this icon when you select the component type as UI.
	Represents the virtual services.
	Represents the dependency between the components.
	Represents the inherited dependencies.

The following table lists the icon that you can use to perform different actions in the system model:

Icon	Used for...
	Adding components, child components, or dependent components.
	Searching for components in the system model.
	Publishing the changes made to the system model, changes made to components, or changes made to component associations. The changes are published to the team space repository.

You can perform the following operations when you view the model that is displayed on the **System Model** page:

Operation	Action
Pan the view of the model	1. Click the area of the model that you want to pan and move the area by keeping the left mouse button pressed. 2. Release the left mouse button to stay on the view that you panned.
Zoom in the view of the model	1. Click the **Zoom-in** icon . The view is enlarged or zoomed in. **Note:** You can repeatedly click the **Zoom-in** icon to further enlarge the view. Alternatively, you can use the scroll button on the mouse or device to zoom in the view.
Zoom out the view of the model	1. Click the **Zoom-out** icon . The view is diminished or zoomed out. **Note:** You can repeatedly click the **Zoom-out** icon to further diminish the view. Alternatively, you can use the scroll button on the mouse or device to zoom out the view.
Reset the view	1. Click the **Menu** icon on the **System Model** page. 2. Select **Reset Zoom**. The view is reset to the default view without any zoom applied. If the view was zoomed in or zoomed out, the view is reset to 100% of the size.
Position the model view	1. Click the **Menu** icon on the **System Model** page. 2. Select from the following options: ◦ **Center**: Select this option to align the view of the model to the center of the pane. ◦ **Fit to View**: Select this option to fit the view of the model in the pane on the **System Model** page.

Tasks in system modeling

When you want to model the system under test for the first time after you install the server software, you must first add the team space repository. See Adding a repository to a team space on page 1031.

After a repository is configured as the team space repository, you can work with the system model. See Tasks for working with a system model on page 1030.

When you want to change, update or delete the repository, you can work with the team space repository. See Tasks for working with the team space repository on page 1031.

Task flows for working with a system model

You can find information about the tasks that you can perform when you want to create a system model, components, associate the components, create linkages among the components, and then publish the system model to the team space repository. You can also find information about the tasks that you must perform when you want to modify an existing system model.

Tasks for working with a system model

You can find the tasks when you want to create a system model for the application under test in your team space.

Tasks	Go to...
Add a repository to the team space.	Adding a repository to a team space on page 1031
Create a system model.	Creating a system model on page 1034
Create components in a system model.	Creating components on page 1036
Create child components to existing components in a system model.	Creating child components on page 1037
Define the relationship between the components in a system model.	Creating linkages between components on page 1039
Delete the relationship between the components in a system model.	Deleting linkages between components on page 1041
Modify components in a system model.	Modifying components on page 1043
Delete components in a system model.	Deleting components on page 1044
Publish changes made to the system model.	Publishing changes to the system model on page 1045
Associate resources with components in a system model.	Associating resources with components on page 1046
View resources associated with components in a system model.	Viewing resources that are associated with components on page 1048

Tasks	Go to...
Remove resources that are associated with components in a system model.	Removing resources that are associated with components on page 1052
Publish changes of associating or dissociating resources in a project with components in the system model.	Publishing changes of associating or dissociating resources with components on page 1053
View system model.	Viewing the system model on page 1055
Delete system model.	Deleting a system model on page 1056

Tasks for working with the team space repository

You can find the tasks that you can perform when you want to work with the team space repository.

Tasks	More information
Add a repository to the team space.	Adding a repository to a team space on page 1031
Delete a team space repository.	Deleting a repository from a team space on page 1056

Adding a repository to a team space

You must add a repository to the team space in Test when you want to create, modify, or store a system model.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* or an *Architect* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you obtained valid credentials to access the Git repository that you want to add.

About this task

As a *Team Space Owner* or an *Architect*, you can add a repository to the team space. You can add a repository to the team space in any of the following methods:

- Add an existing repository that can contain test assets, which are created in the desktop products.
- Add a bare repository.

Attention: The repository that you want to add must not contain any projects or `.project` files under the root directory.

You can add a Git repository using SSH. When you add a repository, the Git repository is cloned to the team space. While you add a repository, you must provide the necessary authentication credentials that are set for the Git repository. If the authentication type is SSH, then you must provide the Git URL, a deploy key, and a passphrase.

For example:

- `git@github.com:account/testRepo.git`
- `ssh://git@10.111.111.333:2222/~/account/project.git`

Important: You can add only one repository to the team space. After you add a repository to the team space, the option to add another repository is not available. You must delete an existing repository before you can add a different repository.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **Manage > Configuration**.
   Result
   The **Configuration** page of the team space is displayed
4. Click **Add Repository**.
   Result
   The **Add Repository** dialog is displayed.
5. Perform the action based on the selected repository type:
   The following table describes the actions required for each repository type:

Repository type	Action
Remote Repository	a. Enter the URL of the Git repository that you want to add to the team space in the **Git URL** field. The required fields are displayed based on the type of Git URL that you entered. b. Enter the required credentials based on any of the following authentication methods configured in the repository. To gain access to the repository, you must use any one of the following authentication methods:

Repository type	Action
	▪ SSH: Deploy key and passphrase ▪ HTTPS: Username and password ▪ HTTP: Username and password **Note:** ▪ You must have defined the authentication type and set the authentication credentials in the Git repository. ▪ If you use SSH to connect to your remote repository and Test displays an `Auth Fail` exception while using the deliver changes option, you can resolve this exception error by regenerating your SSH keys by using the -m PEM option.
Local Repository	Enter a unique repository name in the **Repository Name** field. Optional: Modify the default branch name in the **Default Branch Name** field. By default, the branch name is `main`. **Note:** ◦ You cannot change the Local repository names after creation. ◦ You can create and edit branches, create and edit assets, and publish changes locally in a local repository. ◦ You can later push a local repository to a supported remote repository provider. ◦ Ensure that the branch you are pushing from your local repository does not already exist in the remote repository. This helps avoid merge conflicts.

6. Click **Add**.

Results

The Git repository is added to the team space on Test.

Note: The time to clone the repository can take a few to several minutes and depends on the size of the repository.

What to do next

You can perform the following actions on the repository that you added:

- Update the authentication credentials if they are changed in the Git repository configuration. See Updating the authentication credentials of the repository on page 1745.
- Refresh a repository to fetch and synchronize changes from the remote repository. See Refreshing repositories manually on page 1746.
- Configure a webhook to notify the server if there is a push event in the remote repository. See Creating webhooks on page 1749.
- Add a system model to the repository. See Creating a system model on page 1034.
- Delete a repository that is added to the team space, if it is no longer required. See Deleting a repository from a team space on page 1056.

Related information

Managing repositories on page 1741

Creating a system model

Before you can create components in a system model in Test , you must create an empty system model and publish the system model to the team space repository.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Added a repository to the team space. See Adding a repository to a team space on page 1031.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space for which you want to create a system model.
3. Select an option to create a system model based on your role in the team space or in a project in the team space:

- If you are a *Team space owner* or an *Architect*, go to Step 4 on page 1035.
- If you are not a *Team space owner* or an *Architect* but a project *Owner* or *Tester*, go to Step 5 on page 1035.

4. Perform the following steps, if you are a *Team space owner* or an *Architect*:
   a. Click **System Model** in the left navigation pane.

      The **System Model** page is displayed.

   b. Click the **Add system model** icon .

      An empty system model is created.

   c. Go to Step 6 on page 1035.

5. Perform the following steps, if you are not a *Team space owner* or an *Architect* but a *Project owner*:
   a. Open your project that is displayed under **My projects** tab.

      The **Overview** page of the project is displayed.

   b. Click **System Model** in the left navigation pane.

      The **System Model** page is displayed.

   c. Click the **Add system model** icon .

      An empty system model is created.

   d. Go to Step 6 on page 1035.

6. Click the icon, and then click the **Publish** option.

   Note: If you do not want to commit the changes that you made to the system model, you can select the **Discard** option.

7. Enter a commit message in the **Description of change** field, and then click **Publish**.

   The empty system model is committed and published to the team space repository. When the system model is published to the team space repository, other members of the team space can work with the same system model from any of their projects in the team space.

Results

You have created an empty system model and published it to the team space repository.

What to do next

You can perform any of the following tasks:

- Create components in the system model. See Creating components on page 1036.
- Create child components for the components that you added to the system model. See Creating child components on page 1037.

Creating components

After you create an empty system model in the team space in Test , you can create components that represent components of the system under test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Added a repository to the team space. See Adding a repository to a team space on page 1031.
- Created and published the system model. See Creating a system model on page 1034.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Perform the following steps to add a component to the system model:

   a. Click the **Add component** icon in the **Components** row on the right pane.

      The **Add a component** dialog is displayed.

   b. Enter a name for the component in the **Name** field.

   c. Select the type of component that you want to add from the following options displayed in the **Type** field.

      - **Service**
      - **Database**
      - **UI**

   d. Click **Add**.

The component is displayed as a circular block on the system model pane with the icon and name of the component below the block. The added component is also displayed under the **Components** list on the right pane.

You can create components of other types or the same type in the system model by following the preceding steps.

Results

You have created components in the system model.

What to do next

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.
- Create child components for the components that you added to the system model. See Creating child components on page 1037.
- Create linkages between components in the system model. See Creating linkages between components on page 1039.

Creating child components

You can create child components in the system model in Test when you want to represent the sub-components of a component in a system under test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.

About this task

You can create multiple components in your system model and multiple child components under any component in the system model. After you create a child for a component, the child is represented as a circular block within the circular component block in the system model.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed with the existing components.
4. Create child components by using any of the following options:
   - To create child components from the left pane on the **System Model** page, go to Step 5 on page 1038.
   - To create child components from the right pane on the **System Model** page, go to Step 6 on page 1038.
5. Perform the following steps to create child components from the left pane:
   a. Click the component under which you want to create child components.

      The details of the component are displayed in the right pane.

   b. Click the **Add child** icon in the **Children** row on the right pane.

      The **Add a child to...** dialog is displayed.

   c. Enter a name for the component in the **Name** field.

   d. Select the type of component that you want to add from the following options displayed in the **Type** field.

      - **Service**
      - **Database**
      - **UI**

   e. Click **Add**.

      The child component is displayed as a circular block within the circular block of the parent component on the system model pane. The added child component is also displayed under the **Children** list under the component on the right pane.

      You can create child components of other types or the same type for the same or other components in the system model by following the preceding steps.
6. Perform the following steps to create child components from the right pane of the stem model page:
   a. Click the component name listed under the **Components** in the right pane.

   b. Click the **Add child** icon in the **Children** row on the right pane.

      The **Add a child to...** dialog is displayed.

   c. Enter a name for the component in the **Name** field.

   d. Select the type of component that you want to add from the following options displayed in the **Type** field.

- **Service**
- **Database**
- **UI**

e. Click **Add**.

The child component is displayed as a circular block within the circular block of the parent component on the system model pane. The added child component is also displayed under the **Children** list under the component on the right pane.

You can create child components of other types or the same type for the same or other components in the system model by following the preceding steps.

Results

A component is created and it is associated as a child with the selected component and the name of the child is displayed under the **Children** section on the right pane.

What to do next

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.
- Create linkages between components in the system model. See Creating linkages between components on page 1039.

Creating linkages between components

You can create linkages between components in Test to display the relationship between them. The linkages can be between components or between child components of the same or different parent components.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Created child components. See Creating child components on page 1037.

About this task

You can create multiple components in your system model. You can also create multiple child components under any component in the system model. You can specify the dependency among the components.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Perform the following steps to create a relationship between the components in the system model:

   a. Click the component that you want.

      The details of the component are displayed on the right pane.

   b. Perform the actions in the following table to create dependencies between the components:

Option	Action
Depends on	Perform the following steps, if the component that you selected is dependent on another component: i. Click the **Add dependency** icon in the **Depends on** row on the right pane. The **Add dependency** dialog is displayed. ii. Select the component from the list of components displayed under **Depends on**. iii. Click **Add**. The components linked are listed under the **Depends on** section on the right pane. The linkage between the first to the second component that you selected is shown with an arrow that starts from the first and ends in the second component.
Is a dependency of	Perform the following steps, if the component that you selected has a dependency on another component: i. Click the **Add dependent** icon in the **Is a dependency of** row on the right pane. The **Add dependent** dialog is displayed.

Option	Action
	ii. Select the component from the list of components displayed under the **Is a dependency of** section. iii. Click **Add**. The components linked are listed under **Is a dependency of** section on the right pane. The linkage between the first to the second component that you selected is shown with an arrow that starts from the second and ends in the first component.

Results

You have created linkages between components in the system model.

What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.

Deleting linkages between components

When you do not want to retain the linkages between components that you created in the system model in Test , you can delete the linkages.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Created child components. See Creating child components on page 1037.
- Created linkages between components in the system model. See Creating linkages between components on page 1039.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.

4. Perform the following steps to delete a relationship between the components in the system model:

    a. Click the component that you want.

    The details of the component are displayed on the right pane.

    b. Select from the following types of relationships that you want to delete:

When...	Action
The component depends on another component.	Perform the following steps, if the component that you selected is dependent on another component: i. Select the component from the list of components that are displayed under the **Depends on** section on the right pane. ii. Click the **Delete selected** icon inline with the **Depends on** row. The **Delete relationship** dialog is displayed with the details of the component and the dependent component. iii. Click **Delete**. The component linkage is removed from the system model pane.
The component has a dependency on another component.	Perform the following steps, if the component that you selected has a dependency on another component: i. Select the component from the list of components that are displayed under the **Is a dependency of** section on the right pane. ii. Click the **Delete selected** icon inline with the **Is a dependency of** row. The **Delete relationship** dialog is displayed with the details of the component and the component dependency. iii. Click **Delete**. The component linkage is removed from the system model pane.

Results

You have created linkages between components in the system model.

What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.

Modifying components

At any point in time, after you create a component in Test , you can modify and update the details of the components such as the component type, and the relationships with other components.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Created child components. See Creating child components on page 1037.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Click the component that you want to edit.

   The component details are displayed on the right pane.
5. Perform the actions for the task to modify components as listed in the following table:

Task	Action
Change the component type.	Perform the following steps: a. Identify the component that you want to modify from the components listed under **Components** on the right pane. b. Click the component checkbox. The **Edit selected** icon is enabled. c. Click the **Edit** icon. The **Edit component** dialog box is displayed. d. Select the component type. e. Click **Modify** The type of the component is modified.

Task	Action
Change the name of a component.	Perform the following steps: a. Identify the component that you want to modify from the components listed under **Components** on the right pane. b. Click the component checkbox. The **Edit selected** icon is enabled. c. Click the **Edit** icon. The **Edit component** dialog box is displayed. d. Edit the component name in the **Name** field. e. Click **Modify**. The name of the component is modified.
Change the parent of a component.	Perform the following steps: a. Identify the component that you want to modify from the components listed under **Components** on the right pane. b. Click the component checkbox. The **Change parent** icon is enabled. c. Click the **Change parent** icon. The **Change parent** dialog box is displayed. d. Select another parent for the selected components from those that are listed under the **Parent** list. e. Click **Change**. The parent of the component is changed.

Results

You have edited components and updated the details successfully.

What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.

Deleting components

When you do not require a component in Test , you can delete the component from the system model.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Click the component that you want to delete.

   The component details are displayed on the right pane.
5. Click the **Delete component** icon in the row of the component name.

   The **Delete component** dialog is displayed.
6. Click **Delete**.

Results

You have deleted a component from the system model.

What to do next

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 1045.
- Create components in the system model. See Creating components on page 1036.

Publishing changes to the system model

When you create, delete, or modify components in a system model in Test , the changes are saved locally and are not visible to the other project members or team space members. You must publish the system model changes to the team space repository to make your changes visible to all users of the team space.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 1721 or Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Created child components. See Creating child components on page 1037.
- Created linkages between components in the system model. See Creating linkages between components on page 1039.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Click the icon, and then click the **Publish** option.

   Note: If you do not want to commit the changes that you made to the system model, you can select the **Discard** option.
5. Enter a commit message in the **Description of change** field, and then click **Publish**.

Results

You have published the changes that you made in the system model to the team space repository. Other members of the team space can view and use the system model.

Associating resources with components

After creating components in the system model in Test , you can navigate to a project in the team space to associate the Suites, tests, or virtual service resources in the project repositories with components in the system model.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
4. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
5. Click the component that you want to associate with the Suites, Tests, or virtual service resources.

   The component details are displayed in the right pane.
6. Click the **Associate** tab, and then perform the following steps:

   a. Click **Add**.

   b. Select the type of resource that you want to associate with the component from the following options:

      - Suites and Tests
      - Virtual services

      Note: You can select either of the options or both the options.

      The **Edit branch** dialog is displayed.

   c. Select your action that depends on the existence of an *Edit* branch from the following options:

      - Select an existing *Edit* branch that you created previously from the list of branches listed in the **Select branch** field, and then click **Save**.
      - Click **Create new branch**, enter a name for the branch, and then click **Save**, if you want to create a new branch.

      The **Associate Assets** dialog is displayed. The resources that are contained in the branch of the project repository are listed. The name of the component and the branch are also displayed.

      The **Suites and Tests** option is selected and the assets are listed if you selected the **Suites and Tests** option.

      The **Virtual services** option is selected and the virtual service resources are listed if you selected the **Virtual services** option.

      If you selected both the types of resources, then assets and virtual services in the branch are listed.

   d. Identify the assets or virtual resources that you want to associate with the component in the system model by performing any of the following actions:

- Search for the resource by entering the asset name in the **Search** field.
- Find the assets by scrolling through the list of assets.

e. Select the resources, and then click **Add**.

The selected resources are associated with the component.

You can view the associated resources that are displayed in the **Associate** tab and the **Overview** tab. The test assets are listed under the **Suites and Tests** section and the virtual service resources are listed under the **Virtual services** section.

If you want to associate other resources to the same component, you can add them by clicking the **Add association** icon in the resource row on the right pane. You can then select the resource and add the resource to the selected component.

Results

You have associated Suites, tests, or virtual service resources with a component in the system model.

What to do next

You can perform any of the following tasks:

- Publish the changes made in the system model for associating resources with components. See Publishing changes to the system model on page 1045.

Viewing resources that are associated with components

After you associate Suites, tests, or virtual services resources with components in the system model in Test , you can view the associated resources from the **System Model** page. You can opt to run an instance of the associated resource or stop a running instance of the associated resource.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Associated resources in a project with components in the system model. See Associating resources with components on page 1046.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
4. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
5. Click the component that you associated with the resources in the project repository.

   The resources associated with the component are displayed under the resource type in the **Associate** tab.
6. Click the **Overview** tab, if it is not already displayed.

   The resources that are associated with the component are displayed under the **Suites and Tests** or **Virtual Services** section.
7. View and perform the actions for the resource type that you want from the following options:
   - For **Suites and Tests** resources, go to 8 on page 1049.
   - For **Virtual Services** resources, go to 9 on page 1050.
8. Identify the test resource from those that are listed under the **Suites and Tests** section, and perform the action that you want as described in the following table:

Task	Description	Action
Collapse icon	You can hide the resource sections by clicking the **Collapse** icon. **Note:** When the section is hidden, you can view the number that is displayed of the resources that are associated.	Click the icon to toggle the display to show or hide the resource list in the **Suites and Tests** section.
Type	You can view the type of Suite or test.	Hover over the Suites and Tests icon under the **Type** column to view the type of Suite or test.
Name	You can view the name of the associated Suite or test and the path in the repository.	There is no action to perform.
Status	You can view the details of the test runs.	Click each of the squares to view the details of the specific test run. For example, if the status of a test run is represented as shown in the image , then it indicates that the

Task	Description	Action
		Suite or test has run only two times. After you click any colored square, the **Results** page is displayed.
Action	You can start the execution of the Suite or test from the **Execution** page when you click the **Show in execution page** icon.	Click the **Show in execution page** icon to view the Suite or test on the **Execution** page. The Suite or test is displayed on the **Execution** page.

9. Identify the virtual service resource from those that are listed under the **Virtual Services** section, and perform the action that you want as described in the following table:

Field	Description	Action
Collapse icon	You can hide the resource sections by clicking the **Collapse** icon. **Note:** When the section is hidden, you can view the number that is displayed of the resources that are associated.	Click the icon to toggle the display to show or hide the resource list in the **Virtual Services** section.
Show all icon	You can opt to view the virtual services that are associated with the component either on the **Resources** or **Instances** page. You can start an instance of the virtual service from the **Resources** page while you can stop a running instance from the **Instances** page.	Perform the following steps: a. Click the **Show all** icon. b. Select any of the following actions: ▪ To start an instance of the virtual service: i. Select **Show all in resources page** to go to the **Resources** page. ii. Click the **Execute** icon in the row of the virtual service. iii. Configure the settings for the run of the virtual service, if you want to change any of the settings, else click **Execute.** The virtual service starts to run. ▪ To stop a running instance of the virtual service:

Field	Description	Action
		i. Select **Show all in instances page** to go to the **Instances** page. ii. Identify the instance that you want to stop and click the **Stop** icon in the **Actions** column of the selected virtual service instance. iii. Click **Ok** in the **Stop virtual service instance** dialog that is displayed. The running virtual service instance is stopped.
Name	You can view the name of the associated virtual service and the path in the repository.	There is no action to perform.
Instances	You can view the state of the running instance of the virtual service resource. The state is displayed as `Running` only if the instance is running.	There is no action to perform.
Action	You can go to the **Resources** page to view the virtual service resource when you click the **Show resource** icon . You can start an instance of the virtual service from the **Resources** page.	Click the **Show resource** icon to view the virtual service on the **Resources** page. The virtual service resource is displayed on the **Resources** page.
Expand icon	You can expand the virtual service card only if the virtual service is a running instance. In the expanded card, you can view the details such as the Environment, the total number of requests received by the virtual service, and the relative time	Perform the following steps: a. Expand the virtual service card for a running instance. You can view the details of the instance. b. Click the **Show instance** icon to go to the **Instances** page to view the running instance of the virtual service. The running instance of the virtual service is displayed on the **Instances** page.

Field	Description	Action
	from when the virtual service had started. You can stop a running instance of the virtual service from the **Instances** page.	

Results

You have viewed the resources that are associated with a component from the **System Model** page.

Removing resources that are associated with components

When you do not want to retain the resource that you associated with a component in Test , you can delete the resource association from the system model.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Associated resources in a project with components in the system model. See Associating resources with components on page 1046.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
4. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
5. Click the component that you associated with the resources in the project repository.

   The resources associated with the component are displayed under the resource type in the **Associate** tab.
6. Select the resource that you want to remove from the component by clicking the checkbox of the resource.

   You can select any or all resources for removal.

7. Click the **Delete selected** icon in the row of the resource type.

   The **Disassociate...** dialog is displayed for the type of resource that you selected. For example, the **Disassociate Suites and Tests** dialog is displayed when you selected the test assets.

8. Click **Delete**.

   The **Edit branch** dialog is displayed.

9. Select your action that depends on the existence of an *Edit* branch from the following options:

   - Select an existing *Edit* branch that you created previously from the list of branches listed in the **Select branch** field, and then click **Save**.
   - Click **Create new branch**, enter a name for the branch, and then click **Save**, if you want to create a new branch.

   The dialog that is displayed depends on the type of resource you selected and can be any of the following types:
   - The **Dissociate suites and tests** dialog is displayed, if you selected Suites or tests as the resources to dissociate.
   - The **Dissociate virtual services** dialog is displayed, if you selected the virtual services as the resources to dissociate.

10. Click **Delete**.

Results

You have removed resources that were associated with components in the system model.

What to do next

You must publish the dissociation changes in the system model to the project repository if you want the changes to be reflected in the system model for all members of the project. See Publishing changes of associating or dissociating resources with components on page 1053.

Publishing changes of associating or dissociating resources with components

After you associate or dissociate resources in a project with a component in the system model in Test , you must publish the changes made to the branch in the project repository so that the association changes are visible to the other members of the project.

Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 1034.
- Created components in the system model. See Creating components on page 1036.
- Associated resources in a project with components in the system model. See Associating resources with components on page 1046.
- Dissociated resources from components in the system model. See Removing resources that are associated with components on page 1052.

1. Log in to Test.

   Result

   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Click the icon.

   You can view the options for unpublished changes made to the components and associations in the system model. The options available for association changes are as follows:
   - See unpublished commits
   - Discard
   - Publish
5. Perform any of the following actions to publish the association or dissociation changes:

   - Review the unpublished changes before you publish them. Go to Step 6 on page 1054.
   - Publish the changes. Go to Step 7 on page 1055.
6. Perform the following steps to review the unpublished changes:

   a. Select the **See unpublished commits** option.

      The **Your system model changes** page is displayed. You can find the following information about the changes that you made to the system model when you associated resources with components or deleted the association:
      - The *Edit* branch that you selected to contain the changes is displayed in the **Branch** field. You can select the *Edit* branch that you used before associating resources to components, if it is not already displayed as selected.
      - The projects that contained the resources, which you associated with components in the system model are listed under the **Asset name** column.
      - The type of change is listed under the **Changes** column.
      - The name of the member who made the change is displayed under the **Last changed by** column.
      - The relative time when the change was made is displayed under the **Last updated** column.
      - The number of changes that the *Edit* branch is ahead of from the project repository branch is indicated by the **commits ahead** option. You can view the details by clicking the **commits ahead** option and in the dialog that displays, you can discard the pending commits by clicking the **Reset** option.
      - The number of changes that the *Edit* branch is behind from the project repository branch is indicated by the **commits behind** option. The *Edit* branch can be behind the project repository branch if another member has published or committed other changes in the system model to the same project

repository branch. You can view the details by clicking the **commits behind** option. You can merge all the changes to your *Edit* branch so that the *Edit* branch is updated with all the changes made in the project repository branch.

b. Click **Publish**.

c. Enter a commit message in the **Description of change** field, and then click **Publish**.

Note: If you do not want to commit the changes that you made to the system model, you can select the **Discard** option.

7. Perform the following steps to publish the unpublished changes:

a. Click the **Publish changes** option.

b. Enter a commit message in the **Description of change** field, and then click **Publish**.

Note: If you do not want to commit the changes that you made to the system model, you can select the **Discard** option.

Results

You have published the association changes that you made in the system model to the repository in the project. Other members of the team space can view the association changes and use the system model.

Viewing the system model

You can view an existing system model that is in a team space in Test .

Before you begin

You must be a licensed user of Test and a member of a team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.

Results

You viewed an existing system model in the team space from the **System Model** page.

Deleting a system model

When you want to create a new system model in Test , you must first delete the existing system model, as only one system model is supported per team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **System Model** in the left navigation pane.

   The **System Model** page is displayed.
4. Click the **Delete** icon .

   The **Delete system model** dialog is displayed.
5. Click **Delete**.

Results
You deleted a system model.

What to do next
You must publish the changes to the team space repository. See Publishing changes to the system model on page 1045.

Deleting a repository from a team space

You can delete a repository from a team space in Test when you do not want to use a team space repository.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* or an *Architect* in the team space. See Managing members and their roles in a team space on page 1721.
- Added a repository to the team space. See Adding a repository to a team space on page 1031.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane. Then, use the **Switch to** option to open the team space to which a team space respository is added.
3. Click **Manage > Configuration**.
   Result
   The **Configuration** page of the team space is displayed.

4. Click the **Delete** icon in the action menu list.

   The **Delete repository** dialog is displayed.

5. Select the **I understand...** checkbox, and then click **Delete**.

Results

You have deleted the team space repository.

Restriction: When you delete the team space repository, the system model if it exists is also deleted, and you or other members of the team space cannot add a system model.

Prerequisites to running tests

Before you can run a test asset in a project on Test , you must read the information about the different tests assets. You can add a remote static agent, or Kubernetes cluster to the project and then select it as an alternate location to run the test assets.

You can find the following information about the prerequisite tasks:

- Test run considerations for AFT Suites or Functional tests on page 1057
- Test run considerations for API Suites or API tests on page 1059
- Test run considerations for browser-based Web UI tests on page 1061
- Test run considerations for JMeter tests on page 1061
- Test run considerations for JUnit tests on page 1063
- Test run considerations for Postman tests on page 1066
- Test run considerations for schedules on page 1067
- Test run considerations for Selenium tests on page 1068
- Test run considerations for using mobile device clouds on page 1069
- Management of agents on page 1071
  - Test run considerations for running tests on remote performance agents on page 1073
- Management of remote Kubernetes clusters on page 1101
  - Considerations for using a remote Kubernetes cluster on page 1101
- Management of cloud credentials on page 1115

Related information

Test run configurations on page 1134

Test run considerations for AFT Suites or Functional tests

Before you configure an AFT Suite or a Functional test run in Test , you must first read the considerations that you need to take into account.

When you want to run an AFT Suite or Functional test in a project on Test, you can check if the AFT Suite or Functional test meets any of the following conditions:

If...	Then...
The AFT Suite or Functional test uses the settings configured in an `AFT XML` file for a remote agent	You must ensure that you have configured the location element in the `AFT XML` to point to a remote agent. For more information about setting the location element, refer to Using an XML file to run multiple Web UI tests and Compound Tests simultaneously.
You want to run an AFT Suite or Functional test on a remote agent	You must run the test on the remote agent before you commit the test asset to the remote repository. When you run the test on the remote agent, the following events occur: • The agent is added as the location in the `AFT XML` file, on which the test is to be run. • The agent is displayed as the agent on which the test is to be run, under the **Host** column within the **Location** tab in the **Execute test asset** dialog.
You commit the test asset to the remote repository without running the test on the remote agent	The agent is not displayed under the **Host** column within the **Location** tab in the **Execute test asset** dialog.

Before you configure the run for an AFT Suite or a Functional test, you must complete the following tasks, if they are applicable:

- Add the remote agents, on which the test is to be run, to the project on Test. See Adding a performance agent to a project on page 1076.
- After you add the agent, you can select the agent configured in the `AFT XML` file as the location for the test run or select the remote agents that you add to the project, when you configure a run for an AFT Suite.

  The remote agents are shown as the agents available for selection under the **Override** column within the **Location** tab in the **Execute test asset** dialog.

  You can then select the remote agent as the location to run the AFT Suite or Functional test when you are configuring the test run. Alternatively, you can enter the argument `-swaplocation <configured_agent_location>:<overriding_agent_location>` in the **Program Arguments** field in the **Advanced Setting** dialog when you are configuring the test run.

For running an AFT Suite, see Configuring an AFT Suite run on page 1141.

Related information

Test run configurations on page 1134

Test run considerations for API Suites or API tests

Before you configure an API Suite or API test run, you must first read the considerations that you need to take into account.

You can find the following information about API Suites:

- Important information about API Suites or tests on page 1059
- Important information about report configuration in API Suites or tests on page 1061

Important information about API Suites or tests

When you want to run an API Suite or test from a project on Test and the test asset in HCL DevOps Test Integrations and APIs (Test Integrations and APIs) meets any of the following conditions:

- The Suite or test refers to local stubs that use a transport other than HTTP or MQ.
- The Suite or test refers to local stubs that use the HTTP transport.
- The transport used in the tests or Suite is configured with the hostname set to *localhost* for the HTTP/TCP proxy.
- The Suite or test has tests that use a transport and the transport requires third-party application `Jar` files for a successful run.

You must then refer to the following table for the next steps:

If...	Then...
The test asset refers to local stubs that use a transport other than HTTP or MQ.	You must perform any of the following actions before you commit the test asset to the remote repository: • Remove the reference to the local stub from the test asset. • Publish the stubs to Test Virtualization Control Panel. To publish and edit stubs, see Publishing stubs.
The tests refer to local stubs that use the HTTP transport.	You must perform the following actions before you commit the test asset to the remote repository:

If...	Then...
	1. Create a test or test Suite that contains the stub and tests that run on the stub. 2. Configure the physical HTTP transport with the following parameters in the **Client** tab in the **Physical View**: ◦ Set the `hostname or IP address of the HTTP Proxy server` in the **Proxy Host** field. ◦ Set *3128* as the proxy server port in the **Proxy Port** field. 3. Set the reference to the stub for the tests in the test suite scenario from the **Scenario Editor** dialog from the **Test Factory** view by selecting the *Live system* option in the **Satisfied by** column to run the stubs on the HTTP proxy registered with Test. 4. Save the project.
The transport used in the Suite or tests is configured with the hostname set to *localhost* for the HTTP/TCP proxy.	You must replace the hostname with *fully-qualified-domain-name* or *IPAddress* of the proxy host in the tests before you commit the test asset to the remote repository.
You plan to run the API Suite or test on the computer where Test is installed and the API Suite or test that use a transport from any of the following third-party applications: • Camel • CentraSite • CICS • Coherence • Database • IMS • Integra • JMS • SAP RFC • Software AG Universal Messaging • TIBCO EMS • TIBCO Rendezvous • TIBCO SmartSockets	You must complete the following tasks: 1. Identify the location of the third-party application JAR files. You can use Library Manager to know the location where the third-party application JAR files are saved. 2. Copy the third-party application JAR files to the computer where Test is installed. See Copying third-party application Jars to Kubernetes on page 736.

If...	Then...
• WebSphere Application Server Service Integration Bus (SiBus) • WebLogic • Software AG webMethods • WebSphere MQ	

Important information about report configuration in API Suites or tests

You must configure test Suites or tests in Test Integrations and APIs to use a results database so that the details of the test results can be captured and displayed in the Test API results reports. See Configuring the project results database.

If your server projects use either a Microsoft SQL Server, MYSQL, or both as a project results database, you must copy the Microsoft SQL Server and MySQL database JAR files to the JDBC folder in the path `/data/JDBC` on Test. See Copying third-party application Jars to Kubernetes on page 736.

The database JAR files that you need are based on the version of Microsoft SQL Server or MQSQL that you use with Test Integrations and APIs. See Adding Microsoft SQL Server and MySQL drivers.

To run an API Suite, see Configuring an API Suite or an API test run on page 1174.

Test run considerations for browser-based Web UI tests

Before you configure a Web UI test or a Compound Test that contains Web UI tests, which use either the Chrome or Edge browsers, you must first read the considerations that you need to take into account.

When you want to run a Web UI test or a Compound Test that contains Web UI tests, which use either the Chrome or Edge browser from Test , you must consider the following conditions:

- You must perform any of the following actions:
  - Accept the license agreement of the built-in Chrome and Edge browsers at the time of signing in or logging in to Test.
  - Accept the license agreement of the built-in Chrome and Edge browsers, if it is not accepted at the time of signing in or logging in. See Management of third-party licenses on page 731.

Test run considerations for JMeter tests

Before you can run JMeter tests on Test , you must read the considerations that you must take into account and complete the tasks indicated.

Before you run JMeter tests that you create in JMeter, you must have completed the following tasks:

- You must ensure that the JMeter extension on Test is enabled. You must enable the JMeter extension either at the time of installation of Test or before you add the repository that contains JMeter tests to the server project. See Enabling server extensions on page 740.
- You must have committed the following JMeter assets or resources as a JMeter project to a remote Git repository:

Note: You must have created a project directory and added the associated tests or assets in sub directories, that might be required for the JMeter tests to run successfully.

- The `jmeterRoot.jprj` project file in the project directory.
- The JMeter tests in a sub directory with a name you provide within the project directory.
- Optionally, the Java Key Store that is required at test run time in the `certs` sub directory.
- Optionally, the JMeter properties files that are required at test run time in the `properties` sub directory.
- Optionally, the library jar files that are required at test run time in the `lib` sub directory.
- Optionally, the test dependencies that are required at test run time in the `deps` sub directory.

For example, the project directory can be *myproject* that contains the `jmeterRoot.jprj` project file. In the project directory, you can create the following sub directories:

- The `mytests` sub directory that contains the JMeter tests.
- Optionally, the `certs` sub directory that contains the Java Key Store.

  For example, the path to the key store can be specified in the `system.properties` file as follows:

  ```
  javax.net.ssl.trustStore=$(deploy.home)/certs/simpleStore.jks
  ```

  Where <*simpleStore.jks*> is the Java Key Store and the <*deploy.home*> is the keyword to access the key store at the test run time.
- Optionally, the `properties` sub directory that contains properties files such as `system.properties`, `jmeter.properties`, `reportGenerator.properties`, or `user.properties`.
- Optionally, the `lib` sub directory that contains the **lib** and **lib/ext** content to be used at test run time. For example, the **amq client** and the **plugins** required to parse the **jmx** that contains **amqp** calls.
- Optionally, the `deps` sub directory that contains the test dependencies.

The resulting project directory can be as follows:

```
myproject (that contains the jmeterRoot.jprj)
myproject/mytests
myproject/certs
myproject/properties
myproject/lib
myproject/deps
```

- You must ensure that you use a relative path when JMeter tests refer to other test assets. For example, if the JMeter test that is in the `myproject/mytests` folder refers to a `mycsv.csv` file that is in the `myproject/deps` folder, then the `mycsv.csv` is referred as follows: `../deps/mycsv.csv` in the JMeter test.
- You can create the JMeter project file named as `jmeterRoot.jprj`.

  The project file is a java properties file. The JMeter project file is required so that Test can identify the asset as a JMeter project. You can specify additional information in the project file. You can specify the names of the sub directories created and through keywords you can specify the path in the properties files whether they are relative to the root directory or to the tests sub directory.

For example, if you have created the following sub directories in the project directory called *myproject* that contains the `jmeterRoot.jprj` project file:

- `myproject/mytests`
- `myproject/properties`

If the JMeter tests are contained in the following path: `myproject/mytests/testplan/test1.jmx`, and some dependencies that are also required to be deployed at test run time exists in the following path: `myproject/mytests/testplan/localdir`. You must specify these file paths in the project file as follows:

```
testDir.localtest.sub.dir=localdir
aSecondRootDir.sub.dir=dir2
athirdRootDir.sub.dir=dir3
```

Related reference

Troubleshooting issues on page 1787

Related information

Configuring a run of a JMeter test on page 1377

Test run considerations for JUnit tests

Before you can run JUnit tests on Test , you must read the considerations that you must take into account and complete the tasks indicated.

Before you run JUnit tests, you must have completed the following tasks:

- You must ensure that the JUnit extension on Test is enabled. You must enable the JUnit extension at the time of installation of the server software or before you add the repository that contains the JUnit tests to the server project. See Enabling server extensions on page 740.
- You must have created the JUnit tests that use JDK 8, as part of a Maven V3.6.3 project. The Maven project must contain the Maven Surefire plugin, which is required to run the JUnit tests contained in the Maven project. The Maven project as a `pom.xml` must be committed to the remote repository.

Note: If you create the JUnit tests on other versions of JDK, you must copy the JDK version to the `userlibs` pod under the `/data/junit-ext/jdks` directory. See Copying third-party application Jars to Kubernetes on page 736.

Related reference

Troubleshooting issues on page 1787

Related information

Configuring a run of a JUnit test on page 1386

Test run considerations for using Machine learning capabilities

Before you configure a Performance Test or Schedule run to use Machine learning (ML) capabilities, you must first read the considerations that you need to take into account.

When you want Test to analyze Performance tests or Schedules by using ML capabilities, you must ensure that Performance tests or Schedules conform to the following criteria:

- Must contain HTML tests.
- Must contain stress or performance user profiles.
- Must contain the following minimum stress test requirements:
  - 20 users or more.
  - A minimum of three different user groups and each group has an activity of more than two minutes.
  - Stages that have a reasonably high number of samples within each stage.
  - Stages do not have an increasing number of users.
  - Tests run for longer durations so that a trend can be observed.

You can refer to the following table to view the criteria for analysis of the parameters:

Parameter analyzed	Criteria for analysis
Response Time Lock-Step Pattern	The analyzer identifies the **Response Time Lock-Step Pattern** parameter in the overall page response time observed against the user count based on the following criteria: • Tests have a minimum of 20 users. • Tests run for longer durations for a trend to be observed. • Tests contain a minimum of three different user groups and each group has an activity of more than two minutes.
Response Time Standard Deviation Pattern	The analyzer attempts to detect the response time of pages that are more than thrice the value of the standard deviation calculated for the page response time as the **Response Time Standard Deviation Pattern** parameter.
Throughput Drop Pattern	The analyzer attempts to detect sudden drops in network throughput as the **Throughput Drop Pattern** parameter that is based on the following criteria: **Note:** Sudden drops in throughput might be related to the performance tool itself, issues with network connectivity, or issues with signal-scalability of the system under test. • Tests have a minimum of 20 users. • Tests have stages with a reasonably high number of samples within a stage. • The length and intensity of the throughput drops that are less than 20% of the median observed within a stage are considered by the analyzer. When no

Parameter analyzed	Criteria for analysis
	throughput is received within 60 seconds, such drops are considered in the analysis. Short shark-tooth patterns in drops are not considered by the analyzer. • Parts of the stage with increasing users are not considered for analysis. For example, when the test adds one user every minute and the user range is between one user to 500 users, the **Throughput Drop Pattern** parameter is not analyzed.
Performance Agent Health	The analyzer inspects the health of agents that are participating in a performance execution and identifies potential configuration issues based on the following criteria: • **Minimum requirements** ◦ At least 10 users ◦ At least 10-minute run time ◦ Requires schedule with at least 1 static agent (Performance Agent Health insight is not generated if you are running the test on local host) • **Blocking threads** ◦ Considered as Blocking threads when 90% of available threads are used for 20% of the run time • **Pegged CPU** ◦ Considered as Pegged CPU when 90% of CPU is utilized for 10% of the run time • **Pegged CPU for interval** ◦ Considered as Pegged CPU for interval when there are intervals of high CPU utilization greater than 95% For example, above 95% of CPU utilized for 50% of a 10-minute interval or 98% of CPU utilized over a shorter interval. ◦ Considered as Pegged CPU for interval when there are spikes in CPU utilization during long test runs The agents that are not configured properly can distort performance assessments of the system under test. The analyzer can provide multiple insight types. The insight recommendation texts are parametrized and they provide specific recommendations such as identifying the agents with issues. You can view the following insights: • Pegged CPU for run insight • Pegged CPU for interval insight • Blocked thread insight

Related information

Machine learning capabilities in DevOps Test on page 1543

Test run configurations on page 1134

Test run considerations for Postman tests

Before you can run Postman tests on Test , you must read the considerations that you must take into account and complete the tasks indicated.

Before you run Postman tests that you create in Postman, you must have completed the following tasks:

- You must ensure that the Postman extension on Test is enabled. You must enable the Postman extension at the time of installation of the server software or before you add the repository that contains Postman collections to the server project. See Enabling server extensions on page 740.

  Restriction: If you add a Git repository with Postman collections to a server project before you enable the Postman microservice, the Postman collections might not be displayed in the **Execution** page for you.

- You must have exported the following resources from Postman and saved the resources on your computer:
  - Collections and their associated variables and environments from Postman in which all collections are exported as a single file.

    Note: You can export as many collections as you want.

    For more information, refer to Exporting a collection from Postman and Exporting data dumps.
  - Environments configured in Postman, if you are exporting only the environments as separate JSON files.

    For more information, refer to Exporting environments from Postman.
- You must have committed the following Postman assets or resources to the remote repository:
  - Collections and their associated variables and environments that you exported. You can export all resources that are in your workspace in Postman. You can also export the collections along with the variables as a single file and the environments separately as another file.

    For example, you can save the exported file as `MyCollectionsVarEnv.json` that contains the collections, variables, and environments or you can export only the collections and variables as `MyCollectionsVar.json`.
  - Environments that you exported, if you are exporting only the environments as separate JSON files. For example, you can save the exported file as `MyEnv.json`.
  - An empty text file with the `.postman` extension that is contained in the same directory as the other Postman resources, so that the contents in the directory are attributed as Postman resources. For example, you can create `MyProject.postman`.

Note: The `.postman` file enables Test to identify the test resources as Postman resources.

For example, if you have saved the Postman resources that you downloaded from Postman as follows:

- `MyCollectionsVarEnv.json` or `MyCollectionsVar.json` that contains the Postman collections.
- `MyEnv.json` that contains the environments associated with the Postman collection.

The project directory can be as follows:

```
/Project
/Project/MyProject.postman
/Project/MyCollectionsVarEnv.json
```

Alternatively, the directory can be as follows:

```
/Project
/Project/MyProject.postman
/Project/MyCollectionsVar.json
/Project/MyEnv.json
```

Related reference

Troubleshooting issues on page 1787

Related information

Configuring a run of a Postman test on page 1395

Test run considerations for schedules

Before you configure a Rate Schedule or VU Schedule run, you must first read the considerations that you need to take into account.

Important information about JMeter tests in schedules

When you want to run JMeter tests as part of the VU Schedule or Rate Schedule on Test , you must have completed the following tasks:

- Installed the HCL DevOps Test Performance Agent on a computer.
- Installed the JMeter application on the computer where you have installed the HCL DevOps Test Performance Agent.
- Set the environment variable **JMETER_HOME** that points to the JMeter installation directory.

Important information about Resource Monitoring sources in schedules

When you want to run schedules on Test, you must have completed the following tasks in Test Performance:

- Created a performance schedule of **Rate** or **VU** type.
- Added Resource Monitoring sources, or the labels of Resource Monitoring sources to a schedule. Refer to Adding resource monitoring sources to a performance schedule by using labels.

Before you configure a run of a Schedule, you must complete the following tasks in Test:

- Add Resource Monitoring sources to your project in Test. See Monitoring an NGINX server on page 1566.
- Enable Test as Resource Monitoring service only if you want to replace Resource Monitoring labels set in Test Performance with the labels that you created in Test. See Controlling resource monitoring sources in a schedule on page 1595.
- Perform the following steps when you want to override the Resource Monitoring labels in the test asset with the Resource Monitoring labels that you create in Test:
    1. Click the **Resource Monitoring** tab.

        Note:

        The **Resource Monitoring** tab displays only if you enabled the **Resource Monitoring from Service** option in Test Performance when you created the test asset.

    2. Click , and press the `Ctrl` + `Space bar` keys, or enter the initial letter of a label to select a label in the list.

        Note:

        You can select or add labels only if you added the labels in the **Resource Monitoring Sources** page in Test.

For running a Rate Schedule or VU Schedule, see Configuring a run of a Rate Schedule or VU Schedule on page 1404.

Related information

Test run configurations on page 1134

Test run considerations for Selenium tests

Before you can run Selenium tests on Test , you must read the considerations that you must take into account and complete the tasks indicated.

Before you run Selenium tests, you must have completed the following tasks:

- You must ensure that the JUnit extension on Test is enabled. You must enable the JUnit extension at the time of installation of the server software or before you add the repository that contains the Selenium tests to the server project. See Enabling server extensions on page 740.
- You must have created the Selenium tests that use JDK 8, as part of a Maven V3.6.3 project. The Maven project must contain the Maven Surefire plugin, which is required to run the Selenium tests contained in the Maven project. The Maven project as a `pom.xml` must be committed to a remote Git repository.

  Note: If you create the Selenium tests on other versions of JDK, you must copy the JDK version to the `userlibs` pod under the `/data/junit-ext/jdks` directory. See Copying third-party application Jars to Kubernetes on page 736.

- You must ensure that the following flags are configured in the Selenium test:
  - Firefox is configured as the web browser to run the test.
  - Firefox is enabled to accept all certificates (secured and unsecured).
  - Firefox is configured to not display alerting messages when the test runs in the Firefox browser.
- You must set the system property to point to the Firefox driver (`linux_x64_geckodriver`) that is provided along with the JUnit server extension and is available in the `/tmp` folder.

  For example,

```
System.setProperty("webdriver.gecko.driver", "/tmp/linux_x64_geckodriver")
```

Related reference

Troubleshooting issues on page 1787

Related information

Configuring a run of a Selenium test on page 1417

Test run considerations for using mobile device clouds

Before you configure runs of tests that use mobile device clouds as the location, you must first read the considerations that you need to take into account.

Before you can use mobile devices that are connected to a mobile device cloud from Test , you must complete the following tasks:

- Create any of the following types of tests that are supported to be run from Test.

Test type	Description
AFT Suite	The AFT Suite must specify the path to the tests, device names, and the device cloud location to run the tests.

Test type	Description
	AFT Suites that specify multiple tests or devices on the same device cloud are supported. **Restriction:** AFT Suites that specify a single test on multiple devices or multiple tests on multiple devices when the devices are connected to multiple device clouds are not supported.
Compound Tests that contain mobile tests or Web UI tests that use a device cloud	The Compound Tests must specify multiple mobile tests or multiple Web UI tests that run on a device connected to a single device cloud. **Restriction:** Compound Tests that specify tests on single or multiple devices that are connected to multiple device clouds are not supported.
Individual mobile test or Web UI test that use a device cloud	The individual test must be either a mobile test or a Web UI test that runs on a device connected to a single device cloud.

- Ensure that you have valid credentials for the mobile device clouds, which are supported to be connected from Test. You must have the details as listed in the following table for the mobile device cloud that you want to connect and use:

Attribute	Description	Mobile device cloud				
		BitBar Cloud	**BrowserStack Cloud**	**LambdaTest Cloud**	**pCloudy Cloud**	**Perfecto Mobile Cloud**
Host	The hostname of the mobile device cloud instance.	Required	Required	Required	Required	Required
API key	The API key of your cloud account to authenticate the connection.	Required	Required	Required	Required	Not applicable
Security token	The token used to authenticate the connection.	Not applicable	Not applicable	Not applicable	Not applicable	Required
Project	The project name to use on the mobile device cloud.	Required	Required	Required	Not applicable	Not applicable

Attribute	Description	Mobile device cloud				
		BitBar Cloud	**BrowserStack Cloud**	**LambdaTest Cloud**	**pCloudy Cloud**	**Perfecto Mobile Cloud**
Username	The username of your cloud account.	Not applicable	Required	Required	Required	Not applicable

- Complete the following prerequisite tasks to enable the devices on the mobile device cloud for testing mobile apps or mobile web browsers:

Device cloud	Testing on Android OS	Testing on iOS
BitBar Cloud BrowserStack Cloud Perfecto Mobile Cloud	◦ Uploaded the `.apk` file of the Android application that you want to test. ◦ Installed the Android application that you want to test on the Android device.	◦ Installed the `.ipa` or `.app` file of the iOS application that you want to test. ◦ Installed the iOS application that you want to test on the iOS device.
pCloudy Cloud LambdaTest Cloud	◦ Uploaded the `.apk` file of the Android application that you want to test. ◦ Installed the Android application that you want to test on the Android device. ◦ Verified that the device you want to test is available and in the *Released* state on the cloud platform.	◦ Uploaded the `.ipa` file of the iOS application that you want to test. ◦ Installed the iOS application that you want to test on the iOS device. ◦ Verified that the device you want to test is available and in the *Released* state on the cloud platform.

- Ensure that as a *Project Owner* or *Tester*, you are an authorized user of the cloud credential in your team space. See Viewing details of cloud credentials on page 1126.
- Ensure that you create a variables file that contains the details of the agent or device cloud, if you do not want to use the cloud credentials on the server.

Management of agents

You can find information about managing agents that you install on remote computers and then configure the remote agents to register with Test . After the agents are registered, you can add them to your projects and then select them as locations to run tests or virtual services.

When you plan to run any of the tests from a project on Test on a remote performance agent, see Management of performance agents on page 1072.

When you plan to run any of the supported virtual services from a project on Test on a remote API agent, see Management of API agents on page 1080.

When you plan to run Web UI tests that are created in Test on a remote agent. See Running server test assets on an agent on page 1324.

Management of performance agents

You can manage the performance agents that you want to use as locations to run tests from Test . When you install the agents by using your offline user token, the agents are registered with the team space in Test. You can view the registered agents before you add them to your projects. You can then select them as locations to run tests.

When you plan to run any of the tests on a remote performance agent from a project in Test, you must have completed the following tasks:

- Installed the performance agent of the same version as the version of Test. Refer to Installing DevOps Test Performance Agent.
- Configured the performance agent with your offline user token so that the performance agent can register with Test.

  Note: When you configure the performance agent by using your offline user token, Test considers you as the owner of the performance agent.

- Verified that you have write permissions to the `majordomo.config` file so that certain transactions can be written by Test to the `majordomo.config` file.

  Note: You can verify if you have the *Read* and *Write* permissions by checking the properties of the `majordomo.config` file.

- Viewed the performance agents that are registered with Test. See Viewing performance agents that are registered with DevOps Test on page 1074.
- Added the remote performance agents that are registered with Test to your project on Test. See Adding a performance agent to a project on page 1076.
- After you add the remote performance agents to your project, Test determines the capabilities that are provided by the remote performance agent and displays the details and capabilities of the performance agent on the **Agents and Intercepts** page. You can also define the capabilities that you want to add, and then view, edit, or delete the capabilities that you defined for the performance agents in your project. See Working with the capabilities of a performance agent on page 1078.

Connecting a Docker performance agent to Test

You can run a Performance agent in Docker container and connect it to Test.

Before you begin

You must have the following details to connect the Docker performance agent to Test Hub:

- `TESTHUB_SERVER`: Server host name.
- `TESTHUB_TOKEN`: Offline user token that is obtained in following context.
- `TESTHUB_PORT`: `443`
- `TESTHUB_TEAM_SPACE`: `Initial`
- `TESTHUB_PATH_CONTEXT`: Server API context path.
- `Docker_Image`: Download the `HCL_DT_AGENT_<version_number>_DOCKER_IMG.zip` file from My HCLSoftware and refer to README document in the zip file to load the docker image `hcl-devops-test-performance-<version_number>.tar`.

1. Log in to Test.
2. Click the **User** icon , and select the **Create Token** option.

   Result

   The token is generated automatically and displayed in the **Create Offline User Token** dialog.
3. Click **Copy** to copy the token and save it in a secure location.
4. Run the following command where the Docker image of the performance agent is located:

```
TESTHUB_SERVER=<server_url>
TESTHUB_TOKEN=<access_token>
TESTHUB_PORT=443
TESTHUB_TEAM_SPACE=Initial
TESTHUB_PATH_CONTEXT="/test"
docker run --detach --name Test_performance_agent_docker \
--env TESTHUB_SERVER=$TESTHUB_SERVER \
--env TESTHUB_PORT=$TESTHUB_PORT \
--env TESTHUB_TOKEN=$TESTHUB_TOKEN \
--env TESTHUB_TEAM_SPACE=$TESTHUB_TEAM_SPACE \
--env TESTHUB_PATH_CONTEXT=$TESTHUB_PATH_CONTEXT hcl-devops-test-performance:11.0.9
```

5. Go to **Infrastructure > Agents and Intercepts** and verify whether your agent is connected to Test.
6. Run the following commands to stop and remove the Docker container:

   `docker stop <assigned_name_of_the_container>`

   `docker rm <assigned_name_of_the_container>`
7. Run the following command to view all the help options and available environment variables:

   `docker run --rm hcl-devops-test-performance:11.0.9`

Results

You have connected the Docker performance agent to Test.

Test run considerations for running tests on remote performance agents

Before you configure a test to run on a remote performance agent from Test , you must read the considerations that you must take into account.

Supported tests

You can run the following tests that are supported to be run on a remote performance agent:

- AFT Suite
- Compound Tests that contains performance tests
- Compound Tests that contains Web UI tests
- Compound Tests that contains traditional HTML tests
- Rate Schedule
- VU Schedule

Prerequisite tasks

Before you can run any of the supported tests on a remote performance agent from a project on Test, you must have completed the following tasks:

- Installed the HCL DevOps Test Performance agent on the remote computer. See Management of agents on page 1071.
- Configured the agent with your offline user token so that the agent can connect to Test.
- Added the remote agents to your project on Test. See Adding a performance agent to a project on page 1076.

Note: If you want to run an AFT Suite (that contains Web UI tests) on a performance agent, you must start the performance agent as a non-admin user so that certain browsers that are configured can start correctly during the test run time.

Before you add the tests to the remote repository, you must have completed the following tasks:

- Associated the performance agents with the performance schedule in Test Performance.
- Specified the hostname of the remote workstation in the AFT XML file.

You can continue to configure a run for any of the following supported test types:

- Configuring an AFT Suite run on page 1141
- Configuring a run of a Compound Test that contains performance tests on page 1260
- Configuring a run of a Compound Test that contains Web UI tests on page 1222
- Configuring a run of a Compound Test that contains HTML tests on page 1186
- Configuring a run of a Rate Schedule or VU Schedule on page 1404

Viewing performance agents that are registered with DevOps Test

After you install the remote performance agents and configure them with your offline user token, the performance agents register with Test and you can view the registered performance agents on the **Agents and Intercepts** page.

Before you begin

You must have completed the following tasks:

- Installed the performance agent of the same version as the version of Test.
- Configured the performance agent with your offline user token so that the performance agent can register with Test.

Note: When you configure the performance agent to connect to the server, you must specify the offline user token that you generated on Test. This token defines your ownership over the agent.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test.
3. You can view the agents that you own in any of the following ways:
   - Search for the agent by entering the name of the agent in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case sensitive text that you can enter.

   - Sort the **Type** column to sort the items and display the agents in the rows at the top of the table.
   - Sort the **Agents** column and identify the agents by their names or by their owner.

   You can view the following details about the agents that are registered with Test:
   - The projects to which the agent is added, are displayed in the **Projects** column.
   - The status of the agent is displayed in the **Status** column.

     Note: You can add agents that are in the `Ready` state to your project, and then use them as locations to run tests or virtual services.

Results

You have viewed the registered agents from the **Agents and Intercepts** page.

What to do next

You can perform any of the following tasks:

- You can add the performance agents that you own to your project. See Adding a performance agent to a project on page 1076.
- You can add the API agents that you own to your project. See Adding a performance agent to a project on page 1076.

Adding a performance agent to a project

If you have performance tests that distribute workload across different workstations or Accelerated Functional Testing (AFT) Suites that need to run on different platforms and browsers, you can add HCL DevOps Test Performance Agents to your projects on Test to run tests on remote workstations.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Installed the agent by using your offline user token, which makes you the owner of the agent, or you are assigned the role of a *Team Space Owner*.

About this task

Only you can add or remove a performance agent that you own from the project. However, any member of your project with the permission to run the tests can initiate the run of test assets that are associated with the performance agent.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The **Agents and Intercepts** page is displayed.
3. Click **Add > Add performance agent**.

   The performance agents that you configured with your user token are displayed.

   Note: You can add the performance agents that you own and that are in the *Ready* state. You can add any number of performance agents to your project.

4. Select the performance agents that you want to use in the project, and then click **Add**.

   The performance agents that you added to the project are displayed.
5. Perform the following steps to view details and capabilities of the performance agents:

   a. Expand the performance agent to view the details panel.

      You can find the following details of the performance agent displayed:

Parameter	Description
Version	The version of the installed performance agent.
Address	The IP address of the computer on which the performance agent is installed.
License type	The license type that is enabled for you to use the performance agent.
Load generation service	The status of the performance agent whether it can be used for load generation service.
Last contact	The time of the last contact with the performance agent by Test.

b. Expand the capabilities panel to view the system and application capabilities.

You can find the following capabilities of the performance agent displayed:

Capabilities		Description
System capabilities	Architecture	The architecture of the CPU processor of the computer on which the performance agent is installed.
	MAC Address	The MAC address of the computer on which the performance agent is installed.
	Operating system	The operating system of the computer on which the performance agent is installed.
	RAM	The memory that is allotted to RAM in the computer on which the performance agent is installed.
Application capabilities	Browser	The browsers along with their version that are installed on the computer on which the agent is installed.
User Defined capabilities	User Defined	The capabilities that you add for a performance agent. You can also tag a performance agent so that you can identify the performance agent to select as a location.

c. Check for the status of the performance agent from the **Status** column.

Note: You can add performance agents that are in any state to the project.

Results

You have added remote performance agents to your project and viewed the performance agent capabilities that is displayed for each agent that you own.

What to do next

You can perform any of the following tasks:

- Run the tests on the performance agent that is added to the project from the **Execution** page. See Test run configurations on page 1134.
- Define the capabilities that you want to add, and then view, edit, or delete the capabilities that you defined for the performance agents in your project. See Working with the capabilities of a performance agent on page 1078.

Working with the capabilities of a performance agent

When you installed performance agents V10.1.2 or later, and added them to your project on Test , you can view capabilities of the performance agents from the **Infrastructure** page. When you installed performance agents V10.2.0 or later, you can add, view, edit, or delete properties and capabilities of performance agents from the **User Defined** panel of that performance agent.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Installed the agent by using your offline user token, which makes you the owner of the agent, or you are assigned the role of a *Team Space Owner*.
- Installed a performance agent. See Management of agents on page 1071.
- Added performance agents to your project. See Adding a performance agent to a project on page 1076.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The agents that you added to your project are displayed on the **Agents and Intercepts** page.
3. Expand the performance agent.

   The **Details** and **Capabilities** tabs are displayed.
4. Click **Capabilities** to view the capabilities of the performance agent.
5. Click **User Defined**.

   The **User Defined Capabilities** dialog is displayed.
6. Complete the steps for the task that you want to perform as listed in the following table:

Task	Action
Adding capabilities	Perform the following actions: a. Click **Add new**. b. Enter the name of the capability in the **Capability Name** field.

Task	Action
	c. Perform any of the following actions in the **User Defined Category** field: ▪ Select an existing category from the list. ▪ Enter a name of the category for the capability, if the category does not exist. **Note:** You can also click **Add new** to add a category. The category for the capability is either selected or created and the fields for the property name and value are displayed. **Note:** The property for a capability is a *name-value* pair. d. Enter a name for the property in the **Property Name** field and its corresponding value in the **Property Value** field for the capability. **Note:** You can add multiple properties for the same capability under the same category or under different categories by clicking **Add new property**, and then by entering the name and value for the property. e. Click **Save**. The capabilities that you added are displayed.
Viewing capabilities that you added	The **User Defined Capabilities** panel displays the capabilities that you added to the performance agent.
Editing capabilities that you added	Perform the following actions: a. Click the **Edit** icon in the row of the capability that you want to edit. The user-defined capability is enabled for editing. You can edit the name of the capability, or the property name or property value. b. Edit any of the items that you want for the user-defined capability. c. Click **Update**. The updated capability of the performance agent is displayed.
Deleting capabilities that you added	a. Click the **Delete** icon in the row of the capability that you want to delete. b. Click **Delete**. The capability that you selected is deleted along with all the properties that you added for the capability.

Task	Action
Deleting a property defined for a capability	a. Click the **Edit** icon in the row of the capability that you want to edit and remove a defined property. The user-defined capability is enabled for editing. b. Select and delete the contents in the **Property Name** field and its corresponding **Property Value** field. c. Click **Update**. The property is removed for the capability.

Results

You have added, viewed, edited, or deleted a property or the user-defined capability of a performance agent.

What to do next

You can run the tests on the performance agent from the **Execution** page. See Test run configurations on page 1134.

Management of API agents

You can manage the HCL DevOps Test Integrations and APIs (Test Integrations and APIs) agents that you want to use as locations to run tests from Test . When you install the agents by using your offline user token, the agents are registered with the team space in Test. You can view the registered agents before you add them to your projects. You can then select the agents as locations to run virtual services.

Note: The API agent is referred interchangeably as either the Test Integrations and APIs Agent or API agent in the documentation.

Before you run any of the supported virtual services on a remote Test Integrations and APIs Agent from a project on Test, you must have completed the following tasks:

- Installed the Test Integrations and APIs Agent of the same version as the version of Test. Refer to Installing DevOps Test Integrations and APIs Agent.
- Configured the Test Integrations and APIs agent so that the Test Integrations and APIs agent can register with Test.

  Note: When you configure the Test Integrations and APIs agent by using the offline user token, Test considers you as the owner of the API agent.

- Viewed the API agents that are registered with Test. See Viewing API agents that are registered with DevOps Test on page 1081.
- Added the remote API agents that are registered with Test to your project on Test. See Adding an API agent to a project on page 1083.

If you do not want to retain the API agents in a project or if you want to add the API agents to a different project, you can remove the API agents from a project. When you remove an API agent from a project, you cannot select that API agent to run virtual services in the project. See Removing an API agent from a project on page 1084.

If you want to remove a registered API agent from a team space in Test when the API agent is not available for use, you can unregister a remote API agent from a team space. See Unregistering a remote API agent from DevOps Test on page 1085.

Considerations for running virtual services on remote API agents

Before you configure a run of a virtual service resource on a remote API agent from Test , you must read the considerations that you must take into account.

Supported virtual services

You can run all virtual services that virtualize services on a remote API agent registered with a team space in Test except the following services:

- Virtualized DPL calls in CICS® Transaction Server.
- Virtualized services for IBM® Db2® on IBM® z/OS®.

Prerequisite tasks

Before you run any of the supported virtual services on a remote API agent from a project on Test, you must have completed the following tasks:

- Installed the Test Integrations and APIs Agent on the remote computer. See Management of API agents on page 1080.
- Configured the API agent with your offline user token so that the API agent can connect to Test.
- Added the remote API agents to your project on Test. See Adding an API agent to a project on page 1083.

Licensing

When a virtual service is run on an API agent, the license server configured on the remote agent computer determine the license server that is used by the API agent. This might be different from the license server that is configured for the team space.

Viewing API agents that are registered with DevOps Test

After you install the remote HCL DevOps Test Integrations and APIs (Test Integrations and APIs) agents and configure them with your offline user token, Test Integrations and APIs agents register with Test . You can then view the registered API agents on the **Agents and Intercepts** page.

Before you begin

You must have completed the following tasks:

- Installed the Test Integrations and APIs agent. See Management of API agents on page 1080.
- Configured the Test Integrations and APIs agent so that the Test Integrations and APIs agent can register with Test. See Management of API agents on page 1080.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test.
3. You can view the API agents that you own in any of the following ways:
   - Search for the API agent by entering the name of the API agent in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

   - Sort the **Type** column to sort the items and display the API agents in the rows at the top of the table.
   - Sort the **Agents** column and identify the API agents by their names or by their owners.
4. Perform the following steps to view the details and capabilities of the API agents:
   a. Expand the API agent to view the details panel.

      You can find the following details of the API agent that is displayed:

Parameter	Description
Agent ID	The ID of the installed API agent.
Identifier	The unique identifier of the installed API agent that is configured in the `Agent.config` file for the API agent.
Hostname	The hostname of the computer on which the API agent is installed.
Version	The version of the installed API agent.
Operating system	The operating system of the computer on which the API agent is installed.
Configuration location	The path to the location of the `Agent.config` file on the computer where the API agent is installed.
Tags	The agent attributes or tags that exist in the `Agent.config` file.

   b. Expand the capabilities panel to view the system capabilities.

      You can find the following capabilities of the API agent displayed:

Capabilities	Description
System capabilities	The capabilities of the API agent.

c. Check for the status of the API agent from the **Status** column.

Results

You have viewed the registered API agents from the **Agents and Intercepts** page.

What to do next

You can add the API agents that you own to your project. See Adding an API agent to a project on page 1083.

Adding an API agent to a project

You must add the registered API agents to your projects before you select them as the location to run the virtual service resources that are in your project on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Installed the agent by using your offline user token, which makes you the owner of the agent, or you are assigned the role of a *Team Space Owner*.

About this task

An owner of an API agent or team space owner can add or remove an API agent from the project. However, any member of the project with the permission to run the virtual services can initiate the run of the virtual service resources on API agents that are added to the project.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The **Agents and Intercepts** page is displayed.
3. Click **Add > Add API agent**.

   The API agents that you configured with your user token are displayed.

   Note: You can add the API agents that you own. You can add any number of API agents to your project.
4. Select the API agents that you want to use in the project, and then click **Add**.

The API agents that you added to the project are displayed.

5. Perform the following steps to view the details and capabilities of the API agents:

   a. Expand the API agent to view the details panel.

      You can find the following details of the API agent that is displayed:

Parameter	Description
Agent ID	The ID of the installed API agent.
Identifier	The unique identifier of the installed API agent that is configured in the `Agent.config` file for the API agent.
Hostname	The hostname of the computer on which the API agent is installed.
Version	The version of the installed API agent.
Operating system	The operating system of the computer on which the API agent is installed.
Configuration location	The path to the location of the `Agent.config` file on the computer where the API agent is installed.
Tags	The agent attributes or tags that exist in the `Agent.config` file.

   b. Expand the capabilities panel to view the system capabilities.

      You can find the following capabilities of the API agent displayed:

Capabilities	Description
System capabilities	The capabilities of the API agent.

   c. Check for the status of the API agent from the **Status** column.

Results

You have added remote API agents to your project and viewed the details and capabilities of the API agent that is displayed for each agent that you own.

What to do next

Configure a run of the virtual service resources that are in your project and select an API agent as the location to run the virtual service. See Configuring a run of a virtual service on page 1462.

Removing an API agent from a project

You can remove an API agent from your project in Test when you no longer need the agent to run virtual services.

Before you begin

You must have added the registered API agent to a project in your team space. See Adding an API agent to a project on page 1083.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The agents that you added to your project are displayed on the **Agents and Intercepts** page.
3. You can view the API agents that you own in any of the following ways:
   - Search for the API agent by entering the name of the API agent in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

   - Sort the **Type** column to sort the items and display the API agents in the rows at the top of the table.
   - Sort the **Agents** column and identify the API agents by their names or by their owners.
4. Identify the API agent that you want to remove from the project, and then click the **Remove agent** icon .

   Note: The icon is displayed in the **Actions** column in the row of the API agent only if you are the owner of the remote API agent.

Results

You have removed the registered API agent from your project. The API agent is no longer available as a location to you while you configure a run of virtual service resources that are in your project.

What to do next

You might have to add API agents to your project to use the remote API agents as a location to run tests, if there are no API agents added to your project.

Unregistering a remote API agent from DevOps Test

You can unregister a remote API agent that is registered with Test in a team space when you no longer require it.

Before you begin

You must have completed the following tasks:

- Installed the agent by using your offline user token, which makes you the owner of the agent, or you are assigned the role of a *Team Space Owner*.
- Configured the Test Integrations and APIs agent so that the Test Integrations and APIs agent can register with Test. See Management of API agents on page 1080.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test.
3. Identify the API agent that you want to unregister.

   Note: The icon is displayed in the **Actions** column in the row of the API agent only if you are the owner of the API agent or you are assigned the role of a *Team Space Owner*. The icon is enabled only if the API agent is in a *Lost Contact* state.

4. Unregister the API agent by completing the following steps:
   a. Click the **Unregister API agent** icon .
   b. Click **Unregister** in the **Unregister API agent** dialog box.

Results

You have unregistered the remote API agent from a team space in Test. You cannot run tests on this remote API agent if it is added to your project.

Note: You might have to register it again with Test, if you want to use the remote API agent in your projects.

Management of DevOps Test Runtime agents

When you want to run the server test assets that require resources not available in the cluster, such as the Safari browser, you can use the DevOps Test Runtime agents. You can install the agent on a machine with the required resources, and then select the agent as the location for test runs from Test . You can also manage the registered agents in Test.

When you install the agents by using your offline user token, the agents are registered with the team space in Test, and they are associated with the project that you specified while registering the agents. You can view the registered agents before you add them to the other projects. You can then select the agents as locations to run the server test assets authored in Test.

Before you run any tests on a DevOps Test Runtime agent from a project on Test, you must have completed the following tasks:

- Installed and registered the DevOps Test Runtime agent. See. Installing and registering DevOps Test Runtime agents on page 1087.

  Note: When you configure the DevOps Test Runtime agent by using your offline user token, Test considers you as the owner of the agent.

  Restriction: You can run only one agent on a system.

- Viewed the agents that are registered with Test. See Viewing agents that are registered with DevOps Test on page 1092.
- Added the agents that are registered with Test to your project on Test. See Adding an agent to a project on page 1093.

If the agent is in the **Ready** state, then you can stop the agent by performing the following actions:

1. Switch to the same command line interface where you started the agent.
2. Navigate to the directory where the runtime script is located.
3. Press `Ctrl+C` keys.

If you do not want to retain the agents in a project or if you want to add the agents to a different project, you can remove the agents from a project. When you remove an agent from a project, you cannot select that agent to run tests in the project. See Removing an agent from a project on page 1094.

If you want to remove a registered agent from a team space in Test when the agent is not available for use or in the **Lost Contact** state, you can unregister an agent from a team space. See Unregistering a remote agent from DevOps Test on page 1095.

Installing and registering DevOps Test Runtime agents

When you want to run Web UI tests authored in Test on a Linux, Mac, or Windows system using an agent, you must first install and register the system as an agent by downloading and running the DevOps Test Runtime agent.

Before you begin

You must have completed the following tasks:

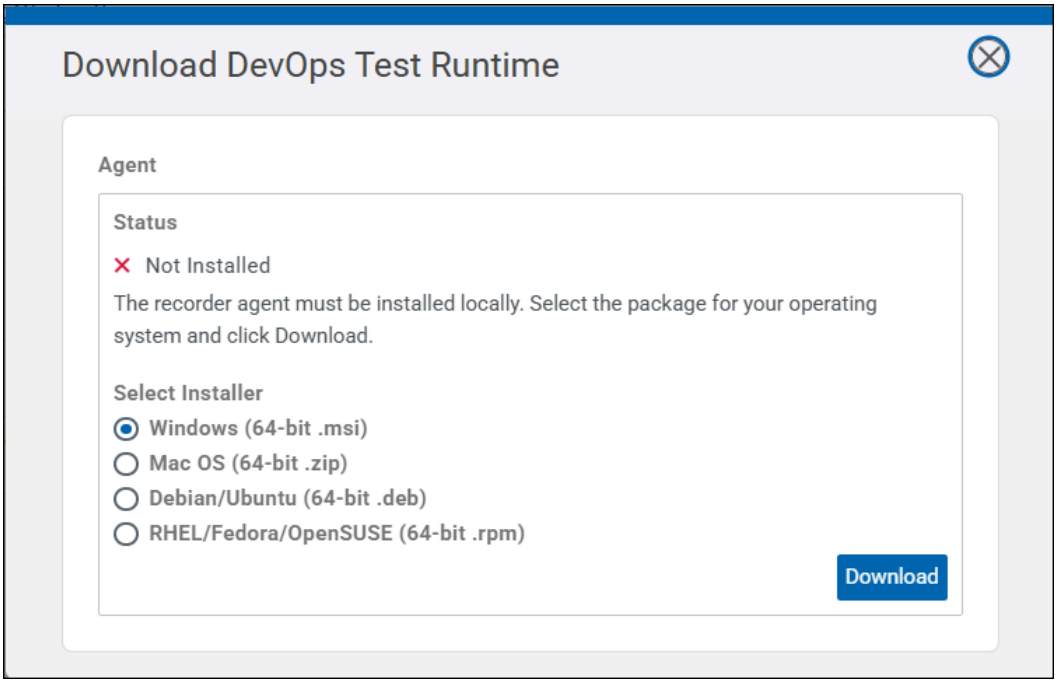
- Ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.
- Generated, copied, and then saved an offline token in a secure location.

1. Log in to Test on the system that you want to register as an agent.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts**.

Alternatively, you can select a project for which the agent should be added and then click **Infrastructure > Agents and Intercepts**.

The **Agents and Intercepts** page is displayed.

3. Click **Download > DevOps Test Runtime**.
   The **Download DevOps Test Runtime** pop-up is displayed.



4. Perform the following steps depending on the operating system:

Operating System	Action
Windows (64-bit)	Perform the following steps to install the agent on Windows: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.msi` file is downloaded. b. Double-click the `.msi` file to install the agent. The installation window with the progress bar is displayed. c. Click **Proceed** to accept the download of the proprietary certificate if prompted. d. Click **Verify** to confirm the agent installation. A confirmation message is displayed after verification is complete. e. Click **Done**. Go to step 5 on page 1090.

Operating System	Action
Mac OS (64-bit)	Perform the following steps to install the agent on Mac: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.zip` file is downloaded. b. Extract the contents of the `.zip` file: unzip devops-test-runtime.zip c. Quarantine the installer: xattr -dr com.apple.quarantine ./* d. Run the installer: ./install.sh e. Click **Verify** to confirm the agent installation. A confirmation message is displayed after verification is complete. f. Click **Done**. Go to step 5 on page 1090.
Debian/Ubuntu (64-bit)	Perform the following steps to install the agent on Ubuntu: a. Click **Download** to download the recording agent installer. The `devops-test-runtime_11.0.9_amd64.deb` file is downloaded. b. Navigate to the directory where the installer is downloaded: cd <download-directory> c. Run the installer: sudo apt install ./devops-test-runtime_11.0.9_amd64.deb d. Click **Verify** to confirm the agent installation. A confirmation message is displayed after verification is complete. e. Click **Done**. Go to step 5 on page 1090.
RHEL/Fedora/OpenSUSE (64-bit)	Perform the following steps to install the agent on Linux: a. Click **Download** to download the recording agent installer. The `devops-test-runtime-11.0.9-1.x86_64.rpm` file is downloaded.

Operating System	Action
	b. Navigate to the directory where the installer is downloaded: cd <download-directory> c. Run the following command: INGRESS_DOMAIN=<domain_name> dnf localinstall ./devops-test-runtime-11.0.9-1.x86_64.rpm d. Click **Verify** to confirm the agent installation. A confirmation message is displayed after verification is complete. e. Click **Done**. Go to step 5 on page 1090.

5. Perform the following steps to register the system as an agent in Test:
   You must provide the following details to register the system as an agent in Test:
   - `<dth_url>`: The URL of Test. For example, https://10.0.2.15.nip.io/test.
   - `<project_id>`: The ID of a project in Test to which the agent must be associated. For example, `2600` is the project ID in https://10.0.2.15.nip.io/test/#/Initial/project/2600.
   - `<offline_token>`: An offline token generated from Test for the logged-in user who is registering the agent to run tests.
   - `<agent_name>`: The name of the agent to be displayed in Test. The name must be unique and must conform to the DNS name format, however, it need not be a real hostname. For example, my-agent.

Operating System	Action

Windows (64-bit .msi)	Perform the following steps:
	a. Navigate to the `C:Users\<user_name>\AppData\HCL\devops-test-runtime` directory where the runtime script is located.
	b. Run the following commands:
	`set DEVOPS_TEST_HUB_URL=<dth_url>`
	`set DEVOPS_TEST_PROJECT_ID=<project_id>`
	`set DEVOPS_TEST_OFFLINE_TOKEN=<offline_token>`
	`set DEVOPS_TEST_AGENT_NAME=<agent_name>`
	`.\devops-test-runtime.exe agent`
	Confirmation messages about the agent starting and registering are displayed. Go to step 6 on page 1091.

Mac OS (64-bit .zip)	Perform the following steps:

Operating System	Action

	a. Navigate to the `~/Applications/devops-test-runtime.app/Contents/Resources` directory where the runtime script is located.
	b. Run the following commands:
	`export DEVOPS_TEST_HUB_URL=<dth_url>`
	`export DEVOPS_TEST_PROJECT_ID=<project_id>`
	`export DEVOPS_TEST_OFFLINE_TOKEN=<offline_token>`
	`export DEVOPS_TEST_AGENT_NAME=<agent_name>`
	`./devops-test-runtime agent`
	Confirmation messages about the agent starting and registering are displayed. Go to step 6 on page 1091.

Debian/Ubuntu (64-bit .deb)	Perform the following steps:
	a. Navigate to the `/opt/hcl/devops-test` directory where the runtime script is located.
	b. Run the following commands:
	`export DEVOPS_TEST_HUB_URL=<dth_url>`
	`export DEVOPS_TEST_PROJECT_ID=<project_id>`
	`export DEVOPS_TEST_OFFLINE_TOKEN=<offline_token>`
	`export DEVOPS_TEST_AGENT_NAME=<agent_name>`
	`./devops-test-runtime agent`
	Confirmation messages about the agent starting and registering are displayed. Go to step 6 on page 1091.

RHEL/Fedora/OpenSUSE (64-bit .rpm)	Perform the following steps:
	a. Navigate to the `/opt/hcl/devops-test` directory where the runtime script is located.
	b. Run the following commands:
	`export DEVOPS_TEST_HUB_URL=<dth_url>`
	`export DEVOPS_TEST_PROJECT_ID=<project_id>`
	`export DEVOPS_TEST_OFFLINE_TOKEN=<offline_token>`
	`export DEVOPS_TEST_AGENT_NAME=<agent_name>`
	`./devops-test-runtime agent`
	Confirmation messages about the agent starting and registering are displayed. Go to step 6 on page 1091.

6. Log in to Test and navigate to **Infrastructure > Agents and Intercepts** to view the agent details that you registered.

Results

You have installed and registered a DevOps Test Runtime agent.

What to do next

You can run the test and view the results. See Running server test assets on an agent on page 1324.

Viewing agents that are registered with DevOps Test

After you install and register the remote DevOps Test Runtime agents with Test , you can view the registered agents on the **Agents and Intercepts** page.

Before you begin

You must have installed and registered the DevOps Test Runtime agents. See Installing and registering DevOps Test Runtime agents on page 1087.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test. The page displays the following columns:
   - **Type**: Type of the agent
   - **Agents**: IP address of the agent with the owner's name
   - **Projects**: Projects that are associated with the agent
   - **Status**: Status of the agent to indicate whether it is **Ready** or **Lost Contact**
   - **Actions**: Actions that are allowed on the agent
3. View the DevOps Test Runtime agents that are registered.
   If the list is long, you can search for the agent by entering the name of the agent in the **Search** field.

   Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

4. Expand the agent to view the details of the agent in four different tabs.
   You can find the following details:

Parameter	Description
DETAILS	
Agent ID	The ID of the installed DevOps Test Runtime agent.
Name	The name that you assigned to the agent during registration.
Hostname	The hostname of the computer on which the DevOps Test Runtime agent is installed.
Address	The IP address of the computer on which the DevOps Test Runtime agent is installed.
Version	The version of the installed DevOps Test Runtime agent.
Operating system	The operating system of the computer on which the agent is installed.
SYSTEM	

Parameter	Description
Architecture	The processor type of the system.
Operating system	The operating system of the computer on which the agent is installed.
RAM	The total RAM size of the system.
APPLICATION	
Browser	Browsers that are installed on the system are listed with the name and version details.

Results

You have viewed the registered DevOps Test Runtime agents from the **Agents and Intercepts** page.

What to do next

You can add the DevOps Test Runtime agents that you own to your project. See Adding an agent to a project on page 1093.

Adding an agent to a project

You must add the registered DevOps Test Runtime agents to your projects before you select them as the location to run the tests.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Installed the agent by using your offline user token, which makes you the owner of the agent, or you are assigned the role of a *Team Space Owner*.

About this task

The owner of a DevOps Test Runtime agent or a team space owner can add or remove an agent from the project. However, any member of the project with the permission to run the tests can initiate the run on agents that are added to the project.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The **Agents and Intercepts** page is displayed.
3. Click **Add > Add DevOps Test Runtime agent**.
   The agents that you configured with your user token are displayed.

Note: You can add the agents that you own. You can add any number of agents to your project.

4. Select the agents that you want to use in the project, and then click **Add**.

   The agents that you added to the project are displayed.

Results

You have added remote agents to your project.

What to do next

You can perform the following tasks:

- View the details and capabilities of the agent that you added. See Viewing agents that are registered with DevOps Test on page 1092.
- Run a Web UI test that is in your project and select an agent as the location to run the test. See Running server test assets on an agent on page 1324.

Removing an agent from a project

You can remove a DevOps Test Runtime agent from your project in Test when you no longer need the agent to run tests.

Before you begin

You must have added the registered DevOps Test Runtime agent to a project in your team space. See Adding an agent to a project on page 1093.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project, and then click **Infrastructure > Agents and Intercepts**.

   The agents that you added to your project are displayed on the **Agents and Intercepts** page.
3. View the DevOps Test Runtime agents that are registered.
   If the list is long, you can search for the agent by entering the name of the agent in the **Search** field.

   Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

4. Identify the agent that you want to remove from the project, and then click the **Remove agent** icon .

   Note: The icon is displayed in the **Actions** column in the row of the agent only if you are the owner of the remote agent.

Results

You have removed the registered agent from your project. The DevOps Test Runtime agent is no longer available as a location to you while you configure a test run in your project.

Unregistering a remote agent from DevOps Test

You can unregister a remote DevOps Test Runtime agent that is registered with Test in a team space when you no longer require it.

Before you begin

You must have installed and registered the DevOps Test Runtime agents, or you are assigned the role of a *Team Space Owner*. See Installing and registering DevOps Test Runtime agents on page 1087.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test.
3. Identify the DevOps Test Runtime agent that you want to unregister.

   Note: The icon is displayed in the **Actions** column in the row of the agent only if you are the owner of the agent or you are assigned the role of a *Team Space Owner*. The icon is enabled only if the agent is in a *Lost Contact* state.

4. Unregister the agent by completing the following steps:
   a. Click the **Unregister DevOps Test Runtime agent** icon .
   b. Click **Unregister** in the **Unregister DevOps Test Runtime agent** dialog.

Results

You have unregistered the remote agent from a team space in Test. As a result, you cannot run tests on this remote agent if it is added to your project.

Management of the personal agent

A personal agent is a local agent that provides a dedicated, isolated runtime environment to run and diagnose Web UI tests and suites directly on your local machine.

A personal agent is a localized DevOps test runtime instance that is linked to your user account. The personal agent is designed for your individual use and remains invisible and inaccessible to other users. For clear identification, each agent is named using a unique `personal:username` format based on your profile credentials. There can be only one active personal agent per user across all machines. When the agent is ready, you can select the **Run on personal agent** option from

the test editor to run the tests locally through that agent instead of a remote container. You can also choose the personal agent as a targeted location to run tests through the **Execute test asset** dialog on the **Execution** page.

The personal agent simplifies and improves the local debugging experience for Web UI tests and suites. As the agent allows you to bypass remote test containers and route tests directly to your local machine, you can visually observe browser behavior in real time and identify the exact point of test failure. This visibility helps you quickly resolve playback errors.

You can use the Personal Agent button located on the test editor page to download, start, or stop the agent. Once you stop the agent, the agent's state changes to inactive in the editor. When the agent is in the inactive state, the **Run on personal agent** option is automatically disabled in the editor interface, and it cannot be used for test runs. As the agent automatically associates with every project you belong to, you can switch between different project workspaces without the need to manually re-associate the agent. You can view the agent listed on the **Agents & Intercepts** page both inside a project and in the initial team space. To ensure privacy and prevent shared access, a different user in the same project can neither view the agent on the **Agents & Intercepts** page nor select it as a target location to run tests.

Installing and starting the personal agent

When you want to run server test assets or server suite assets locally on your system and view the live test run, you can install and use the personal agent. To run tests locally, you must first download the DevOps Test Runtime agent, and then install and start the agent in **Test Editor**.

Before you begin

You must have ensured that you are a Team Space Owner, Project Owner, or Member with a Tester role to create or edit a test resource.

1. Log in to Test on the system that you want to register as an agent.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > project_name** to open the project that contains the test assets.
   The **Overview** page of the project is displayed.
3. Click **Author > Test Editor**.
   The **Test Editor** page and the test navigator panel are displayed.
4. Click the button located in the page title bar next to the branch selector.
   The **Manage DevOps Test Runtime** dialog opens and checks the status of your local installation.
5. Perform one of the following steps depending on the installation status of the agent:

If...	Then...
The agent is not installed, and the screen displays the following options: ◦ Windows (64-bit .msi) ◦ Mac OS (64-bit .zip)	Perform the following steps to install the agent on Windows: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.msi` file is downloaded. b. Double-click the `.msi` file to install the agent.

If...	Then...
◦ Debian/Ubuntu (64-bit .deb) ◦ RHEL/Fedora/OpenSUSE (64-bit .rpm)	The installation window with the progress bar is displayed. c. Click **Proceed** to accept the download of the proprietary certificate if prompted. d. Click **Verify** to confirm the agent installation. A browser notification might get displayed for you to confirm the opening of the agent. You must click **Open Link**. The **Launch** button is enabled. Go to step 6 on page 1099.
	Perform the following steps to install the agent on Mac: a. Click **Download** to download the recording agent installer. The `devops-test-runtime.zip` file is downloaded. b. Extract the `.zip` file: unzip devops-test-runtime.zip c. Quarantine the installer: xattr -dr com.apple.quarantine ./* d. Run the installer: ./install.sh e. Click **Verify** to confirm the agent installation. The **Launch** button is enabled. Go to step 6 on page 1099.
	Perform the following steps to install the agent on Ubuntu: a. Click **Download** to download the recording agent installer. The `devops-test-runtime_11.0.9_amd64.deb` file is downloaded. b. Navigate to the directory where the installer is downloaded: cd <download-directory> c. Run the installer: sudo apt install ./devops-test-runtime_11.0.9_amd64.deb d. Click **Verify** to confirm the agent installation. The **Launch** button is enabled. Go to step 6 on page 1099.
	Perform the following steps to install the agent on Linux:

<table>
<tr><th>If...</th><th>Then...</th></tr>
<tr><td></td><td>a. Click Download to download the recording agent installer.

The <code>devops-test-runtime-11.0.9-1.x86_64.rpm</code> file is downloaded.
b. Navigate to the directory where the installer is downloaded:

cd <download-directory>
c. Run the following command:

INGRESS_DOMAIN=<domain_name> dnf

localinstall ./devops-test-runtime-11.0.9-1.x86_64.rpm
d. Click Verify to confirm the agent installation.

The Launch button is enabled. Go to step 6 on page 1099.</td></tr>
<tr><td rowspan="2">The agent is installed, but not updated, and the screen displays the following options:
◦ Windows (64-bit .msi)
◦ Mac OS (64-bit .zip)
◦ Debian/Ubuntu (64-bit .deb)
◦ RHEL/Fedora/OpenSUSE (64-bit .rpm)</td><td>Perform the following steps to update the agent on Windows:
a. Click Upgrade.

The installation screen with the progress bar is displayed. The latest DevOps Test Runtime is installed and configured.
b. Click Proceed to accept the download of the proprietary certificate if prompted.

The Launch button is enabled. Go to step 6 on page 1099.</td></tr>
<tr><td>Perform the following steps to install the agent on Mac:
a. Click Download to download the recording agent installer.
b. Uninstall the existing agent:

rm -rf ~/Applications/devops-test-runtime.app
c. Extract the downloaded <code>.zip</code> file:

unzip devops-test-runtime.zip
d. Quarantine the installer:

xattr -dr com.apple.quarantine ./*
e. Run the installer:

./install.sh
f. Click Verify to confirm the agent installation.

The Launch button is enabled. Go to step 6 on page 1099.</td></tr>
</table>

If...	Then...
	Perform the following steps to install the agent on Ubuntu: a. Click **Download** to download the recording agent installer. b. Uninstall the existing agent: sudo apt remove devops-test-runtime c. Navigate to the directory where the installer is downloaded: cd <download-directory> d. Run the following command: sudo apt install ./devops-test-runtime_11.0.9_amd64.deb e. Click **Verify** to confirm the agent installation. The **Launch** button is enabled. Go to step 6 on page 1099.
	Perform the following steps to install the agent on Linux: a. Click **Download** to download the recording agent installer. b. Uninstall the existing agent: sudo yum remove devops-test-runtime c. Navigate to the directory where the installer is downloaded: cd <download-directory> d. Run the installer: sudo INGRESS_DOMAIN=<domain_name> dnf localinstall ./devops-test-runtime-11.0.9-1.x86_64.rpm e. Click **Verify** to confirm the agent installation. The **Launch** button is enabled. Go to step 6 on page 1099.
The agent is installed.	The **Launch** button is displayed. Go to step 6 on page 1099.

6. Click **Launch**.

   The personal agent button changes to , and the tooltip displays the following message when hovered over: `Personal Agent Ready - Click to stop`. This message indicates that the agent is active.

Results

You have downloaded and installed a DevOps Test Runtime agent. The personal agent is listed and visible only to you when you navigate to **Infrastructure > Agents and Intercepts**, both in the initial team space and in the projects that you are associated with.

What to do next

You can run the Web UI tests or suites and view the results. See Running server test assets on page 1321 and Running server suite assets on page 1329.

Stopping or unregistering the personal agent

You can stop or unregister the personal agent when you no longer need the agent to run tests.

Before you begin

You must have added the personal agent in your team space, and it must be in the active state. See Installing and starting the personal agent on page 1096.

Perform the steps by using one of the following two methods:

- Stop the agent through the **Test Editor** page.
  a. Log in to Test on the system that you want to register as an agent.

     The **Projects** page of the initial team space is displayed.

  b. Click **My projects > project_name** to open the project that contains the test assets.

     The **Overview** page of the project is displayed.

  c. Click **Author > Test Editor**.

     The **Test Editor** page and the test navigator panel are displayed.

  d. Click the button located in the page title bar next to the branch selector.

     The **Stop DevOps Test Runtime** dialog opens.

  e. Click **Yes** to stop the agent.

     The personal agent button changes to , and the tooltip displays the following message when hovered over:

     `Personal Agent Inactive - Click to launch` . This message indicates that the agent is inactive.

- Unregister the agent through the **Agents and Intercepts** page.
  a. Open the **Initial Team Space** page and then navigate to the **Infrastructure > Agents and Intercepts** page.
  b. View the personal agent that is listed with the personal:username name.

     If the list is long, you can search for the agent by entering the name of the agent in the **Search** field.

Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

c. Identify the agent, and then click the **Remove agent** icon .

The **Unregister DevOps Test Runtime** dialog opens.

d. Click **Unregister** to remove the agent.

Results

You have stopped or unregistered the personal agent. The agent will not be available as a location to run tests till you start it again.

Management of remote Kubernetes clusters

You can find information about the tasks that you can perform on remote Kubernetes clusters. After you set up a remote Kubernetes cluster, you can register the external cluster with a team space on Test . You must then add the remote cluster to a project before you run test assets or resources on the remote Kubernetes cluster.

You can find the following information about managing the external Kubernetes cluster:

- Considerations for using a remote Kubernetes cluster on page 1101
- Registering a remote Kubernetes cluster on page 1104
- Viewing remote Kubernetes clusters on page 1107
- Adding a remote Kubernetes cluster to a project on page 1109
- Editing configurations of a remote Kubernetes cluster on page 1110
- Removing a remote Kubernetes cluster from a project on page 1113
- Unregistering a remote Kubernetes cluster from a team space on page 1114

Considerations for using a remote Kubernetes cluster

When you want to use a remote Kubernetes cluster as a location to run the supported tests or test resources, you must read the considerations first.

Before you want to run any of the supported tests or test resources on Test that are configured to run on a remote Kubernetes cluster, you must have completed the following tasks:

- Verified that the Kubernetes cluster that you want to use is a supported one by referring to the OS, platform, and containers that are specified for Test in System Requirements for DevOps Test 2026.06 (11.0.9).
- Kept a suitable `kubeconfig` file with access to the remote Kubernetes cluster in a folder on the computer that you want to use to access an instance of Test.

You must be familiar with the administration of a Kubernetes cluster. When you as the cluster administrator, set up the remote Kubernetes cluster must consider the following requirements:

- You must have a `kubeconfig` file with access to the remote Kubernetes cluster in a folder on the computer that you want to use to access an instance of Test.

  Note: If the `kubeconfig` file references certificate files by using filepaths then the certificate data must be embedded in the `kubeconfig` file to make it usable by Test. If the `kubeconfig` file contains multiple contexts, then only the current context is required.

  You can use the following kubectl command to create a suitable `kubeconfig` file called `portable-config` in the current folder. The `portable-config` file contains embedded certificate data and the information used by the current context:

  ```
  kubectl config view --flatten --minify > portable-config
  ```

- You must ensure that the remote Kubernetes cluster is able to access the Test image registry which is the same registry that you used to install Test. There is no need to copy the Test image to the remote Kubernetes cluster but the remote cluster must be able to access the image registry at test runtime.

  If the remote cluster that you want to use is a Lightweight Kubernetes (k3s) cluster, it might require a *Secret* to pull the Test image. You must run the following command on the remote k3s cluster:

  ```
  kubectl create secret docker-registry hclcr.io \
  -n {remote-namespace} \
  --docker-server=hclcr.io \
  --docker-username={okta-email-address} \
  --docker-password={harbor-cli-secret} \
  --docker-email={example@abc.com}
  ```

  Notes:
  - You must replace *{remote-namespace}* with the value of the *namespace* that you have used for the remote cluster.
  - You must replace *{okta-email-address}* with the username of the Harbor repository and replace *{harbor-cli-secret}* with the secret key that you copied from the Harbor repository.
  - If the username contains any special characters, such as *$*, you must enclose it within single quotation-marks.
  - You can replace *{example@abc.com}* with the email address of the administrator, if required.

- You need not configure separate licenses for running on the remote cluster because the licenses are consumed from the pool of available licenses that are configured for the team space on Test. The licenses are returned to the pool when the test run is completed on the remote cluster.
- You must ensure that the Test domain is accessible from the remote cluster so that the pod in the cluster is able to communicate with Test.
- You must ensure that the appropriate **PriorityClass** exists on the remote cluster if you want to control the priority given to the remote execution pod by the scheduler. The same **PriorityClass** that is used by the Test execution pod is used by the remote execution pod. If the named **PriorityClass** does not exist in the remote cluster, then no **PriorityClass** is specified by the remote execution pod.

- You must create an HTTP or an HTTPS Emissary listener when you want to run virtual services on the remote cluster. You must create both listeners when you want to run virtual services that use both HTTP and HTTPS transports. For information about creating listeners, refer to https://www.getambassador.io/docs/emissary/latest/topics/running/listener. The Emissary listener associates with the hosts created by Test in the target namespace for executions that require Ingress. For example, when you want to run virtual services on the remote cluster and run tests from the remote cluster that use static agents.
- You can enter the code snippets for *nodeSelectors* and *tolerations* in a `yaml` format, if you want to use *nodeSelectors* and *tolerations* on the remote cluster. You can enter these in the fields provided in the **Advanced** configuration dialog when you register a remote cluster with your team space on Test.

  For more information about *nodeSelector*, see nodeSelector and for *tolerations*, see Tolerations.

Restrictions: The following restrictions apply when you want to run the supported tests on a remote Kubernetes cluster:

- Using the **UserLibs** that are required for running certain API Suites are not supported on the remote cluster.
- Running of Compound Tests that contain performance tests, which require the **NodePort** access to connect to static agents, are not supported on the remote cluster.

When you plan to run any of the test assets or resources on a remote Kubernetes cluster from a project on Test, you must complete the following tasks:

	Tasks	More information...
1	Register a remote Kubernetes cluster with a team space on Test.	Registering a remote Kubernetes cluster on page 1104
2	View remote Kubernetes clusters added to a team space.	Viewing remote Kubernetes clusters on page 1107
3	Add a remote Kubernetes cluster to a project in a team space.	Adding a remote Kubernetes cluster to a project on page 1109
4.1	Configure a run for your test that you want to run on the remote Kubernetes cluster by performing the following actions: 1. Check for the prerequisite tasks that you must perform before you configure a test run for the test. 2. Select the task for the test that you want to configure for a run, and then follow the steps to run the test.	1. Test run configurations on page 1134 2. Prerequisites to running tests on page 1057 3. Test run configurations on page 1134

	Tasks	More information...
4.2	Configure a run of the virtual service resources that you want to run on the remote Kubernetes cluster by performing the following actions: 1. Check for the prerequisite tasks that you must perform before you configure a run of the virtual services. 2. Select the virtual service resources in your project that you want to configure for a run, and then follow the steps to run them.	1. Prerequisites for running HTTP virtual services on page 1455 2. Configuring a run of a virtual service on page 1462

Related information

Test run configurations on page 1134

Management of remote Kubernetes clusters on page 1101

Registering a remote Kubernetes cluster

You must register a remote Kubernetes cluster to your team space on Test before you add the remote cluster to your project.

Before you begin

You must have completed the following tasks:

- Verified that the Kubernetes cluster that you want to use is a supported one by referring to the OS, platform, and containers that are specified for Test in System Requirements for DevOps Test 2026.06 (11.0.9).
- Kept a suitable `kubeconfig` file with access to the remote Kubernetes cluster in a folder on the computer that you want to use to access an instance of Test.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane of the team space.

   The **Agents and Intercepts** page is displayed.
3. Click **Add new > Add new cluster**.

   The **New cluster** dialog is displayed.

4. Perform the following steps to add the `kubeconfig` file:

    a. Click **Browse**, and locate the `kubeconfig` file of the Kubernetes cluster that you want to add.

    b. Select the `kubeconfig` file, and then click **Open**.

    The name of the `kubeconfig` file that you selected is displayed along with the address of the Kubernetes cluster that is specified in the file.

    c. Click **Next**.

The address of the Kubernetes cluster that is configured in the `kubeconfig` file is displayed in the **Address** field.

Note: You can change the address of the Kubernetes cluster if it is different from the address in the `kubeconfig` file.

5. Perform the following steps to add a namespace and name for the cluster:

    a. Enter the namespace of the cluster in the **Namespace** field.

    b. Enter a name for the cluster in the **Name** field.

    The name helps you to identify the cluster.

    c. Click **Next**.

6. Perform the following steps to add the ingress details of the cluster:

    a. Select the ingress type of the cluster from the **Ingress type** list.

    You can select the ingress type from the following options that are displayed:

Option	Description	Result
None	When selected specifies that no ingress type is configured for the remote Kubernetes cluster.	You can run any of the supported tests on the remote cluster but not on the remote agents.
NGINX	When selected specifies that NGINX is configured as the ingress type for the remote Kubernetes cluster.	You can run any of the supported tests on the remote cluster or on any of the remote agents.

    b. Enter the ingress host pattern of the cluster in the **Ingress host pattern** field.

    The DNS used by a computer that runs a client application (such as a static agent) or the HTTP proxy that routes traffic to the virtual service needs to resolve such hostnames to the IP address of the ingress domain.

    For example, if the IP of the remote cluster is 12.12.12.123, then you must enter the ingress host pattern as **.12.12.12.123.nip.io*

The *Unique ID* of the virtual service is prefixed to the ingress hostname when virtual services are routed from the Test cluster.

c. Enter the name of the secret in the remote namespace, which contains the certificate information.

You must have created the secret and named it following the naming conventions per the Kubernetes guidelines. The certificate that you create is a wildcard certificate which contains the Ingress host pattern of the Kubernetes cluster. This wildcard certificate ensures that the client applications can trust the virtual services.

Note: If a secret name is not provided then the default certificate that is provided by NGINX is used, which might cause issues to authenticate the virtual service by the client applications.

d. Click **Next**.

7. Optionally, perform the following steps to enter the code snippets for *nodeSelectors* and *tolerations*, if applicable for the remote cluster:

a. Enter the code snippet for *nodeSelectors* in a `yaml` format in the **Node selectors** field.

b. Enter the code snippet for *tolerations* in a `yaml` format in the **Tolerations** field.

c. Select the **Copy pull secret to the cluster** checkbox, if it is not already selected.

Note: The remote Kubernetes cluster pulls the secret to access the Test image registry which is the same registry that you used to install Test. The remote Kubernetes cluster must be able to access the image registry at test runtime.

d. Click **Register**.

The remote cluster is added to the team space and is displayed with the icon in the **Type** column on the **Agents and Intercepts** page. You can identify the cluster with its name in the **Agents** column.

Note: The **Projects** column does not display any projects that are associated with the cluster yet.

Results

You have registered a remote Kubernetes cluster with your team space on Test.

What to do next

You must add the remote cluster to your project in your team space before you use the remote cluster as a location to run test assets or resources in your project. See Adding a remote Kubernetes cluster to a project on page 1109.

Related information

Management of remote Kubernetes clusters on page 1101

Viewing remote Kubernetes clusters

You can view the clusters that you have registered with a team space on Test to view their details or edit their configurations.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Registered a remote Kubernetes cluster with your team space on Test. See Registering a remote Kubernetes cluster on page 1104.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane of the team space.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts, or clusters that are registered with the team space on Test.
3. You can view the clusters that you own in any of the following ways:
   - Search for the cluster by entering the name of the cluster in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

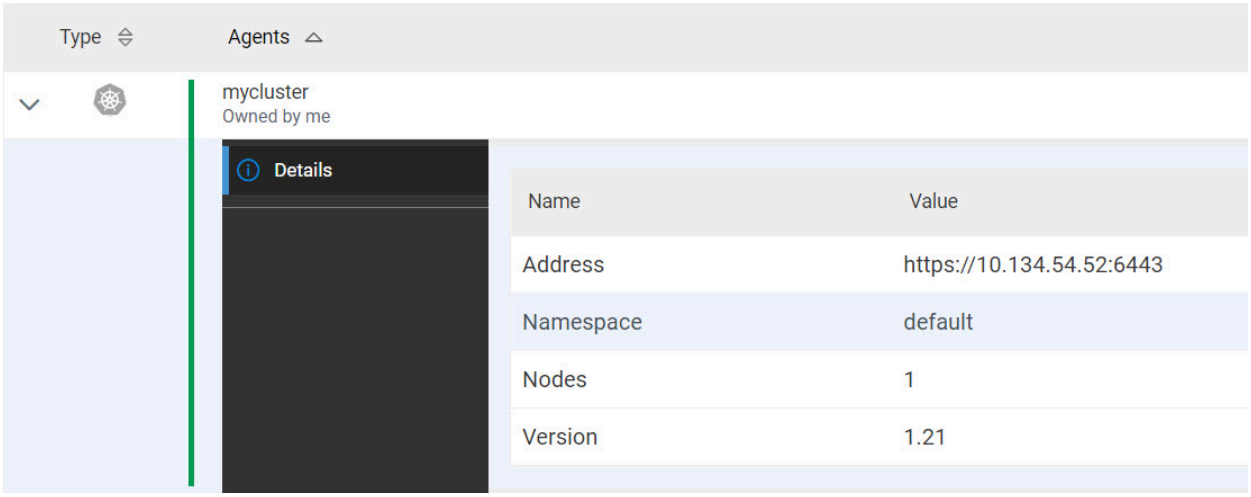
   - Sort the **Type** column to sort the items, and then identify the cluster by the name or owner.
   - Sort the **Agents** column to sort the items, and then identify the cluster by the name or owner.

   You can view the following details about the clusters that are in the team space:

- The projects in the team space to which the cluster is added are displayed in the **Projects** column.
- The status of the cluster is displayed in the **Status** column.

Note: You can add clusters that are in the `Ready` state to your project, and then use them as locations to run tests.

You can view the details of the cluster by clicking the **Expand** icon.



The following table describes the details about the cluster:

Name	Description
Address	Specifies the address of the remote Kubernetes API.
Namespace	Specifies the *namespace* of the remote Kubernetes cluster in which the test runs occur.
Nodes	Specifies the number of nodes in the remote Kubernetes cluster.
Version	Specifies the Kubernetes version of the remote Kubernetes cluster.

Results

You have viewed details of the remote clusters that are registered with a team space on Test.

What to do next

You must add the remote cluster to your project in your team space before you can use the remote cluster as a location to run test assets or resources that are in your project. See Adding a remote Kubernetes cluster to a project on page 1109.

Related information

Management of remote Kubernetes clusters on page 1101

Adding a remote Kubernetes cluster to a project

You must add a remote Kubernetes cluster to your project after you register the cluster with your team space on Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Registered a remote Kubernetes cluster with your team space on Test. See Registering a remote Kubernetes cluster on page 1104.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Been assigned a role as an *Owner* in the project.
- Created a project.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open your project.
   The **Overview** page is displayed.
3. Click **Infrastructure > Agents and Intercepts** in the navigation pane of the team space.

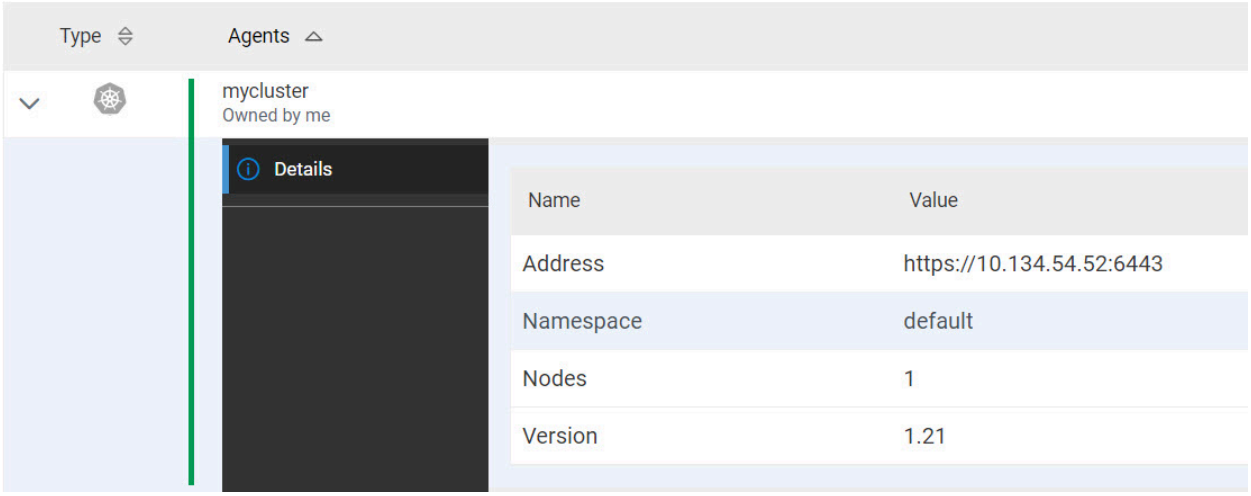
   The **Agents and Intercepts** page is displayed.
4. Click **Add > Add cluster**.

   The **Add cluster** dialog is displayed.
5. Select the cluster that you want to add to the project from the list of clusters that are registered with the team space, and then click **Add**.

   Note: You can add the clusters that are registered with the team space. You can add any number of clusters to your project.

   The clusters that you added to the project are displayed.

   You can view the details of the cluster by clicking the **Expand** icon ›.



The following table describes the details about the cluster:

Name	Description
Address	Specifies the address of the remote Kubernetes API.
Namespace	Specifies the *namespace* of the remote Kubernetes cluster in which the test runs occur.
Nodes	Specifies the number of nodes in the remote Kubernetes cluster.
Version	Specifies the Kubernetes version of the remote Kubernetes cluster.

What to do next

You can select any of the remote clusters as an alternate location to run the test assets or resources that are in your project while you configure a run from the **Execution** page.

Related information

Test run configurations on page 1134

Management of remote Kubernetes clusters on page 1101

Editing configurations of a remote Kubernetes cluster

You can edit the `kubeconfig` file, address, namespace, name, or ingress details of the remote Kubernetes cluster that you added to a team space on Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Registered a remote Kubernetes cluster with your team space on Test. See Registering a remote Kubernetes cluster on page 1104.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane of the team space.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts, or clusters that are registered with the team space on Test.
3. You can view the clusters that you own in any of the following ways:
   - Search for the cluster by entering the name of the cluster in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

   - Sort the **Type** column to sort the items, and then identify the cluster by the name or owner.
   - Sort the **Agents** column to sort the items, and then identify the cluster by the name or owner.

   You can view the following details about the clusters that are in the team space:
   - The projects in the team space to which the cluster is added, are displayed in the **Projects** column.
   - The status of the cluster is displayed in the **Status** column.

     Note: You can add clusters that are in the `Ready` state to your project, and then use them as locations to run tests.

4. Identify the cluster that you want to modify the configurations, and then click the **Edit** icon .

   Note: The icon is displayed in the **Actions** column in the row of the cluster only if you own the cluster.

   The **Update cluster** dialog is displayed.
5. Perform the following steps to change the `kubeconfig` file:
   a. Click **Browse**, and locate the `kubeconfig` file of the Kubernetes cluster that you want to use.

   b. Select the `kubeconfig` file, and then click **Open**.

The name of the `kubeconfig` file that you selected is displayed along with the address of the Kubernetes cluster that is specified in the file.

c. Click **Next**.

6. Perform the following steps to change the namespace or name of the cluster:
   a. Edit or change the namespace of the cluster that you want to use, in the **Namespace** field.
   b. Edit or change the name of the cluster in the **Name** field.
   c. Click **Next**.
7. Perform the following steps to change the ingress details of the cluster:

   a. Select the ingress type of the cluster from the **Ingress type** list, if you want to use a different cluster.

   b. Enter or change the ingress host pattern of the cluster in the **Ingress host pattern** field, if you did not provide it when setting up the remote cluster or change it to match the cluster you want to use.

   c. Enter the secret if you did not provide it when setting up the remote cluster.
   The secret is used to authenticate the remote cluster to use the licenses from the Team space License Server so that the inbound traffic is allowed into the ingress domain.

   Note: The secret to use must be available with your Test server administrator.

   d. Click **Next**.

8. Optionally, perform the following steps to edit or change the code snippets for *nodeSelectors* and *tolerations*, if applicable for the remote cluster:
   a. Enter the code snippet for *nodeSelectors* in a `yaml` format in the **Node selectors** field.
   b. Enter the code snippet for *tolerations* in a `yaml` format in the **Tolerations** field.
9. Click **Update**.

   The remote cluster is displayed with the updated configurations.

   Note: The projects that are associated with the cluster are displayed in the **Projects** column and you need not add the cluster to the same project again.

Results

You have updated an existing remote cluster with the changed configurations.

What to do next

You can select the updated cluster as an alternate location to run the test assets or resources that are in your project while you configure a run from the **Execution** page.

Related information

Management of remote Kubernetes clusters on page 1101

Removing a remote Kubernetes cluster from a project

You can remove a remote Kubernetes cluster from a project in a team space on Test when you no longer need the cluster to run tests.

Before you begin

You must have completed the following tasks:

1. Registered a remote Kubernetes cluster with your team space on Test. See Registering a remote Kubernetes cluster on page 1104.
2. Added the Kubernetes cluster to a project in your team space. See Adding a remote Kubernetes cluster to a project on page 1109.

   Note: You can remove remote Kubernetes clusters that only you added to the project.

3. Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
4. Been assigned a role as an *Owner* in the project.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project in your team space.

   The **Overview** page is displayed.
3. Click **Infrastructure > Agents and Intercepts** in the navigation pane of your team space.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts, or clusters that are added to your project.
4. You can view the clusters that you own in any of the following ways:
   - Search for the cluster by entering the name of the cluster in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

   - Sort the **Type** column to sort the items, and then identify the cluster by the name or owner.
   - Sort the **Agents** column to sort the items, and then identify the cluster by the name or owner.

5. Identify the cluster that you want to remove from the project, and then click the **Remove cluster** icon .

   Note: The icon is displayed in the **Actions** column in the row of the cluster only if you own the remote cluster.

   A message is displayed that the remote cluster is removed from the project successfully.

Results

You have removed the remote cluster from your project. The remote cluster is no longer available as a location when you configure a run of the test assets or resources that are in your project.

What to do next

You might have to add a remote cluster to your project to use the remote cluster as a location to run test assets or resources.

Related information

Management of remote Kubernetes clusters on page 1101

Unregistering a remote Kubernetes cluster from a team space on page 1114

Editing configurations of a remote Kubernetes cluster on page 1110

Unregistering a remote Kubernetes cluster from a team space

You can unregister a remote Kubernetes cluster from a team space in Test when you no longer require it.

Before you begin

You must have completed the following tasks:

1. Registered a remote Kubernetes cluster with your team space on Test. See Registering a remote Kubernetes cluster on page 1104.
2. Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
3. Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane of the team space.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts, or clusters that are registered with the team space on Test.

3. You can view the clusters that you own in any of the following ways:
    - Search for the cluster by entering the name of the cluster in the **Search** field.

      Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

    - Sort the **Type** column to sort the items, and then identify the cluster by the name or owner.
    - Sort the **Agents** column to sort the items, and then identify the cluster by the name or owner.
4. Identify the cluster that you want to remove.

   Note: The and icons are displayed in the **Actions** column in the row of the cluster only if you own the remote cluster.

5. Unregister the cluster by completing the following steps:
    a. Click the **Unregister cluster** icon .
    b. Click **Unregister** in the **Unregister cluster** dialog box.

   A message is displayed that the remote cluster is unregistered successfully.

Results

You have unregistered the remote cluster from a team space on Test. You cannot run test assets or resources on this remote cluster if it is added to your project. You might have to add it again to your team space, if you want to use the remote cluster in your projects.

Related information

Management of remote Kubernetes clusters on page 1101

Editing configurations of a remote Kubernetes cluster on page 1110

Adding a remote Kubernetes cluster to a project on page 1109

Removing a remote Kubernetes cluster from a project on page 1113

Management of cloud credentials

You can find information about managing the credentials that are required to connect to cloud-based applications such as mobile device clouds and AI providers from Test .

Contents

- Mobile device clouds on page 1116
- Mobile device clouds and Test on page 1116
- Mobile device clouds that are supported on page 1116

- Details required of mobile device clouds on page 1116
- Prerequisites for using mobile devices on mobile device clouds on page 1117
- Overview of tasks to manage mobile device clouds on page 1118
- AI providers on page 1118
- Details required of AI providers on page 1119

Mobile device clouds

Mobile device clouds are cloud computing services offered by cloud service providers who host real mobile devices in a cloud. You can purchase plans from the service providers to use the mobile devices in the cloud to test your mobile apps. The cloud service providers also offer different browsers in the cloud to test web applications.

To use the cloud services, you must subscribe to their services that suit your testing requirements.

Mobile device clouds and Test

If you already use mobile device clouds to test your mobile apps, testers in your organization must be required to set up the connection to the mobile device clouds individually. They use the same credentials that you subscribed to as an organization. This can be a potential risk of exposure of your subscription details and without proper monitoring can lead to misuse of the credentials.

You can use Test to connect to mobile device clouds. You must first configure a cloud credential by using the details of your cloud subscription account.

You can then add testers from your organization as authorized users who can use the credential to connect to mobile device clouds. Such members need not enter the subscription details to connect to the cloud. Only the authorized users of the credential can run their tests on the devices that are connected to the mobile device cloud. With this mechanism, you can prevent the exposure of the credentials of your subscription to testers and also prevent testers from creating separate connections to the same mobile device cloud.

Mobile device clouds that are supported

Test supports connections to the following mobile device clouds:

- BitBar Cloud
- BrowserStack Cloud
- LambdaTest Cloud
- pCloudy Cloud
- Perfecto Mobile Cloud

Details required of mobile device clouds

You must ensure that you have the details and valid credentials of your subscription to the mobile device cloud that you want to connect and use. See the following table for the details required for the different mobile device clouds:

Attribute	Description	Mobile device cloud				
		BitBar Cloud	**BrowserStack Cloud**	**LambdaTest Cloud**	**pCloudy Cloud**	**Perfecto Mobile Cloud**
Host	The hostname of the mobile device cloud instance.	Required	Required	Required	Required	Required
API key	The API key of your cloud account to authenticate the connection.	Required	Required	Required	Required	Not applicable
Security token	The token used to authenticate the connection.	Not applicable	Not applicable	Not applicable	Not applicable	Required
Project	The project name to use on the mobile device cloud.	Required	Required	Required	Not applicable	Not applicable
Username	The username of your cloud account.	Not applicable	Required	Required	Required	Not applicable

Prerequisites for using mobile devices on mobile device clouds

When you want to use the available mobile devices that are connected to a mobile device cloud, you must ensure that you have completed certain tasks. Refer to the following table for details of the tasks that must be completed before tests can be run on the mobile devices connected to mobile device clouds:

Device cloud	Testing on Android OS	Testing on iOS
BitBar Cloud BrowserStack Cloud Perfecto Mobile Cloud	• Uploaded the `.apk` file of the Android application that you want to test. • Installed the Android application that you want to test on the Android device.	• Installed the `.ipa` or `.app` file of the iOS application that you want to test. • Installed the iOS application that you want to test on the iOS device.
pCloudy Cloud LambdaTest Cloud	• Uploaded the `.apk` file of the Android application that you want to test. • Installed the Android application that you want to test on the Android device. • Verified that the device you want to test is available and in the *Released* state on the cloud platform.	• Uploaded the `.ipa` file of the iOS application that you want to test. • Installed the iOS application that you want to test on the iOS device. • Verified that the device you want to test is available and in the *Released* state on the cloud platform.

Overview of tasks to manage mobile device clouds

You as a *Team Space Owner* can perform the following tasks:

- Configure the connection to mobile device clouds. You must create a *cloud credential* in the team space that represents a connection to a mobile device cloud. You must use the details of your mobile device cloud subscription account to configure the cloud credential to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Add your team space members to access the mobile device cloud. The added members become the authorized users of the cloud credential, and then can connect to the mobile device cloud from Test. See Adding authorized users to a cloud credential on page 1124.
- View the details of the cloud credential that you added to the team space. See Viewing details of cloud credentials on page 1126.
- Remove the authorized users of the cloud credential, when you want to reduce the number of users accessing the mobile device cloud. See Removing authorized users from a cloud credential on page 1132.
- Edit an existing cloud credential that you configured to modify the name of the credential, project, or username of your cloud subscription account. See Editing details of cloud credentials on page 1129.
- Delete an existing cloud credential that you configured, from the team space. You might want to delete the credential when your cloud subscription to the mobile device cloud has expired and you no longer need the connection. See Deleting an existing cloud credential on page 1133.
- Export details of all existing cloud credentials that are configured in your team space, when you might want to use the details elsewhere. You can export details such as the name of the credentials, the credential type, and the email ID of authorized users of the credentials. See Viewing details of cloud credentials on page 1126.

You as a *Project Owner* or *Tester* can perform the following tasks on cloud credentials or the mobile device clouds:

- Ensure that you are added as an authorized user of a cloud credential in the team space. See Viewing details of cloud credentials on page 1126.
- Configure the following mobile tests to run on the mobile devices that are connected to mobile device clouds:
  - Configuring an AFT Suite to run on a mobile device by using a cloud credential on page 1166.
  - Configuring a run of a Compound Test that contains mobile tests to run by using a cloud credential on page 1212.

AI providers

AI providers are the organizations that build, develop, and offer AI technologies. They offer models and tools that power artificial intelligence in everything from virtual assistants to complex data analysis systems. You can purchase plans from the AI providers to use the services in the cloud or locally to generate Web UI tests. For the cloud-based provider such as OpenAI, you must subscribe to their services. For the locally hosted provider such as Ollama, you must download the software and set up a server.

Test supports the following AI providers:

- OpenAI
- Ollama
- IBM watsonx
- Anthropic

Details required of AI providers

You must ensure that you have the details and valid credentials of the AI account that you want to connect and use. The following table lists the details required for the different accounts:

Attribute	Description	AI Agents			
		OpenAI	**Ollama**	**IBM watsonx**	**Anthropic**
API End Point	The URL of the AI provider.	Optional	Required	Required	Optional
API key	The API key of the AI provider account to authenticate the connection.	Required	Optional	Required	Required
Project Id	A unique identifier assigned to the project that you have created.	Not applicable	Not applicable	Required	Not applicable
Model	The name of the large language model (LLM) that you want to use for test generation.	Not applicable	Optional	Required	Optional

Adding cloud credentials of mobile device clouds

You as a *Team Space Owner* can add a cloud credential in the team space by using the details of your mobile device cloud subscription. This cloud credential represents a connection between Test and a mobile device cloud.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you have the details of your subscription and valid credentials of the mobile device cloud to which you want to connect. See Details required of mobile device clouds on page 1116.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed.
3. Click **Add cloud credential**.

The **Details** tab of the **Add cloud credential** dialog is displayed.

4. Perform the following steps in the **Details** tab:

    a. Enter a name in the **Name** field so that you can identify the mobile device cloud, which is associated with a specific subscription account.

    b. Select the mobile device cloud from the **Cloud Authentication** list.

    c. Click **Next**.

    The **Authentication** tab of the **Add cloud credential** dialog is displayed.

5. Enter the relevant details of your subscription account that you have in the mobile device cloud, in the **Authentication** tab.

    You can find information about the details that are required for each of the supported mobile device clouds and the action you need to perform for the required fields in the following table:

Field	**Mobile device cloud**					**Action**
	BitBar Cloud	**BrowserStack Cloud**	**LambdaTest Cloud**	**pCloudy Cloud**	**Perfecto Mobile Cloud**	
Host	Required	Required	Required	Required	Required	Enter the hostname of the mobile device cloud instance.
API key	Required	Required	Required	Required	Not applicable	Enter the API key of your cloud subscription account to authenticate the connection.
Security token	Not applicable	Not applicable	Not applicable	Not applicable	Required	Enter the token used to authenticate the connection.
Project	Required	Required	Required	Not applicable	Not applicable	Enter or select the project to use that is in the mobile device cloud.
Username	Not applicable	Required	Required	Required	Not applicable	Enter the username of your cloud subscription account.

6. Click **Add**.

    The cloud credential for your specific subscription account with a mobile device cloud is displayed in a table on the **Cloud Credentials** page.

    You can find the following details about the cloud credential on the **Cloud Credentials** page.

<table>
<tr><th>Column header</th><th>Details displayed</th></tr>
<tr><td>Provider</td><td>The name of the mobile device cloud service provider.

You can view the Toggle icon associated with the service providers when you add a cloud credential.

Note: If you are the owner of the cloud credential, you can switch to an On/Off position. However, when you are not the owner and you try to use the Toggle icon an error message is displayed.</td></tr>
<tr><td>Name</td><td>The name you entered for the cloud credential when you created the connection in the team space.</td></tr>
<tr><td>Credential Type</td><td>The type of cloud that is connected to Test. Displays Device Cloud when you create a credential to connect to a mobile device cloud.</td></tr>
<tr><td>Members</td><td>The abbreviated names of the team space members who are authorized users of the cloud credential. The abbreviated name is the combination of the first letters of the first name and last name.</td></tr>
<tr><td>Actions</td><td>The icons for the different actions that you can perform on the credential. Refer to the following table for the icons and the actions:

<table>
<tr><th>Icon</th><th>Action</th><th>Available for</th></tr>
<tr><td></td><td>Click to view details of the credential of the cloud subscription account.</td><td>Team space owner and authorized user</td></tr>
<tr><td></td><td>Click to edit and update the credential for any of your cloud subscription details.</td><td>Team space owner</td></tr>
<tr><td></td><td>Click to remove the configured connection to the mobile device cloud account from the server.</td><td>Team space owner</td></tr>
<tr><td></td><td>Click to add team space members as authorized users of the credential.</td><td>Team space owner</td></tr>
</table>
</td></tr>
</table>

Results

You have added credentials of mobile device cloud accounts so that Test can connect to the mobile device clouds.

What to do next

You can add team space members as authorized users of the credentials. See Adding authorized users to a cloud credential on page 1124.

Related information

Management of cloud credentials on page 1115

Adding cloud credentials of AI providers

You can add a cloud credential in the team space by using the details of your AI provider accounts so that Test can connect to the providers.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you have the details of your subscription and valid credentials of the cloud account to which you want to connect. See Details required of AI providers on page 1119.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed.
3. Click **Add cloud credential**.

   The **Details** tab of the **Add cloud credential** dialog is displayed.
4. Perform the following steps in the **Details** tab:
   a. Enter a name in the **Name** field so that you can identify the mobile device cloud, which is associated with a specific subscription account.
   b. Select an AI provider from the **Cloud Authentication** list.
   c. Click **Next**.

      The **Authentication** tab of the **Add cloud credential** dialog is displayed.
5. Enter the relevant details of your subscription account for AI providers in the **Authentication** tab.

You can find information about the details that are required for each of the supported AI providers in the following table:

Attribute	Description	AI Agents			
		OpenAI	**Ollama**	**IBM watsonx**	**Anthropic**
API End Point	The URL of the AI provider.	Optional	Required	Required	Optional
API key	The API key of the AI provider account to authenticate the connection.	Required	Optional	Required	Required
Project Id	A unique identifier assigned to the project that you have created.	Not applicable	Not applicable	Required	Not applicable
Model	The name of the large language model (LLM) that you want to use for test generation.	Not applicable	Optional	Required	Optional

6. Click **Add**.

The cloud credential for your specific subscription account is displayed in a table on the **Cloud Credentials** page.

You can find the following details about the cloud credentials on the **Cloud Credentials** page.

Column header	Details
Provider	The name of the AI provider. You can view the Toggle icon associated with the service providers when you add a cloud credential. **Note:** If you are the owner of the credential, you can switch to an On/Off position. However, when you are not the owner and you try to use the Toggle icon, an error message is displayed.
Name	The name you entered for the credential when you created the connection in the team space.
Credential Type	The type of account that is connected to Test. Displays *AI Provider* when you create a credential to connect to an AI account.

Column header	Details
Members	The abbreviated names of the team space members who are authorized users of the cloud credential. The abbreviated name is the combination of the first letters of the first name and last name.
Actions	The icons for the different actions that you can perform on the credential. Refer to the following table for the icons and the actions:

Icon	Action	Available for
	Click to view details of the credential.	Team space owner and authorized user
	Click to edit and update the credential.	Team space owner
	Click to remove the configured connection to the AI provider account from the server.	Team space owner
	Click to add team space members as authorized users of the credential.	Team space owner

Results

You have added credentials of AI provider accounts in Test.

What to do next

You can add team space members as authorized users of the credentials. See Adding authorized users to a cloud credential on page 1124.

Related information

Management of cloud credentials on page 1115

Adding authorized users to a cloud credential

After you add the cloud application account details to a cloud credential in your team space inTest , you can add other team space members as authorized users of this cloud credential. The authorized users can then use the connected cloud application account for tests.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Added the cloud credentials of your mobile device cloud or AI provider account to configure the connection between Test and the cloud account. See Adding cloud credentials of mobile device clouds on page 1119 and Adding cloud credentials of AI providers on page 1122.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed with the cloud credentials that are added to the team space.
3. Identify the cloud credential by performing any of the following actions:
   - Scroll through the table of cloud credentials to identify the cloud credential with the name you provided.
   - Search for the cloud credential by entering the name of the credential in the **Search** field.
   - Sort the **Provider** column so that the names of the cloud providers are listed alphabetically in either increasing or decreasing order. Identify your cloud provider and look for the name of your credential.

   You identified the cloud credential that you want to use.
4. Perform the following steps to add team space members as authorized users:
   a. Click the **Add Members** icon in the row of the credential.

      The **Add New Member** dialog is displayed.

   b. Enter any text that is contained in the name or email ID of the team space member that you want to add in the field.

      The team space users are listed whose names or email ID matches the search text.

   c. Select the team space member.

      The selected member is added to the list of authorized users and is displayed under **Members with access**.

   d. Click **Close** to close the dialog.

      The abbreviated name of the team space member is displayed under the **Members** column in the row of the cloud credential.

      Note: The abbreviated name is the combination of the first letters of the first name and last name.

      You can view the first name, last name, username, and the email ID of the added member when you hover over the abbreviated name.

Results

You as a *Team Space Owner* added team space members as authorized users of the cloud credential.

What to do next

You can inform the team space members that they are added as authorized users of the cloud credential. The authorized users can then use the credential to connect to the cloud account from Test and run or generate tests. The authorized users of the credential can manage the following tasks:

- Configuring an AFT Suite to run on a mobile device by using a cloud credential on page 1166.
- Configuring a run of a Compound Test that contains mobile tests to run by using a cloud credential on page 1212.
- Configuring a run of a Compound Test that contains Web UI tests to run by using a cloud credential on page 1249
- Configuring a run of a mobile test to run on a mobile device by using a cloud credential on page 1286
- Configuring a run of a Web UI test to run by using a cloud credential on page 1354
- Running server test assets by using a cloud credential on page 1326
- Running server suite assets by using a cloud credential on page 1331

Related information

Management of cloud credentials on page 1115

Viewing details of cloud credentials

If you are a *Team Space Owner* in Test , you can view the details of the cloud credential when you want to decide your action for adding or removing team space members, changing your cloud subscription details, or deleting a credential. If you are a *Project Owner* or *Tester*, you can verify whether you are added as an authorized user to use the cloud credential.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space, a *Project Owner*, or a *Tester* in the project. See Managing members and their roles in a team space on page 1721.
- Added the cloud credentials of your mobile device cloud or AI provider account to configure the connection between Test and the cloud account. See Adding cloud credentials of mobile device clouds on page 1119 and Adding cloud credentials of AI providers on page 1122.
- Added team space members as authorized users of the cloud credential. See Adding authorized users to a cloud credential on page 1124.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed with the cloud credentials that are added to the team space.
3. Identify the cloud credential by performing any of the following actions:

- Scroll through the table of cloud credentials to identify the cloud credential with the name you provided.
- Search for the cloud credential by entering the name of the credential in the **Search** field.
- Sort the **Provider** column so that the names of the cloud providers are listed alphabetically in either increasing or decreasing order. Identify your cloud provider and look for the name of your credential.

You identified the cloud credential that you want to use.

4. Select the action that you want to perform:
   - Verify whether you are added as an authorized user of the cloud credential by the *Team Space Owner*, if you are a *Project Owner* or *Tester*, go to Step 5 on page 1127.
   - View details of the cloud credential, if you are a *Team Space Owner*, *Project Owner*, or *Tester*, go to Step 6 on page 1127.
   - Export the details of all the cloud credentials in your team space, if you are a *Team Space Owner*, go to Step 7 on page 1129.
5. Perform the following steps to verify whether you are added as an authorized user of the cloud credential:
   a. Look for your abbreviated name under the **Members** column in the row of the identified cloud credential.

      Note: The abbreviated name is the combination of the first letters of the first name and last name.

   b. Hover over the abbreviated name to view the first name, last name, username, and the email ID to verify that they are your details.

      You have verified whether you are an authorized user of the cloud credential.

      Note: If you do not find your abbreviated name or details, you need to contact your team space owner to add you as an authorized user.

   c. Go to Results on page 1129.
6. Click the **View** icon in the row of the credential that you want to view.

   The details of the cloud credential are displayed in the **View cloud credential** dialog.

   You can view the following details of the mobile device cloud or AI provider that was configured in the cloud credential:

Attribute	Description	Mobile device cloud or AI provider details								
		BitBar Cloud	BrowserStack Cloud	LambdaTest Cloud	pCloudy Cloud	Perfecto Mobile Cloud	OpenAI	Ollama	IBM watsonx	Anthropic
Owner	The username and email ID of the team space owner who configured the cloud credential.	✔	✔	✔	✔	✔	✔	✔	✔	✔
Credential name	The name provided to the cloud credential by the team space owner when the credential was configured.	✔	✔	✔	✔	✔	✔	✔	✔	✔
Type	The name of the mobile device cloud or AI provider to which the connection was configured in the cloud credential.	✔	✔	✔	✔	✔	✔	✔	✔	✔
Host	The hostname or URL of the mobile device cloud instance.	✔	✔	✔	✔	✔	✘	✘	✘	✘
Project	The name of the project that you created in your cloud subscription account.	✔	✔	✘	✘	✘	✘	✘	✔	✘
Username	The username of your cloud subscription account.	✘	✔	✔	✔	✘	✘	✘	✘	✘
API End Point *	The URL of the AI provider.	✘	✘	✘	✘	✘	✘	✔	✔	✔
Model	The name of large language model (LLM) that you want to use for test generation.	✘	✘	✘	✘	✘	✘	✔	✔	✔
API key	The API key of your cloud subscription account to authenticate the connection.	✔	✔	✔	✔	✘	✔	✔	✔	✔
Security token	The token used to authenticate the connection.	✘	✘	✘	✘	✔	✘	✘	✘	✘

Attribute	Description	Mobile device cloud or AI provider details								
		BitBar Cloud	BrowserStack Cloud	LambdaTest Cloud	pCloudy Cloud	Perfecto Mobile Cloud	OpenAI	Ollama	IBM watsonx	Anthropic
Access to	The team space members that the team space owner added as authorized users of the cloud credential.	✔	✔	✔	✔	✔	✔	✔	✔	✔

You can close the dialog by clicking **Close**.

7. Click the **Download CSV** option to export details of all existing cloud credentials that are configured in your team space when you want to use the details elsewhere, such as the name of the credentials, the credential type, and email ID of authorized users of the credentials members.

   You saved the details of all the credentials in your team space as a CSV file to your computer.

Results

You have completed the following tasks:

- If you are a *Project Owner* or *Tester*, you verified whether you are an authorized user of the cloud credential.
- If you are a *Team Space Owner*, *Project Owner*, or *Tester*, you have viewed the details of the cloud credential that you added.
- If you are a *Team Space Owner*, you saved the details of all cloud credentials that are in your team space to a file on your computer.

Related information

Management of cloud credentials on page 1115

Editing details of cloud credentials

If any of the details of your cloud subscription account change, as a *Team Space Owner* in Test you can view the details of the existing cloud credential and edit it to update the credential with the latest details.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Added the cloud credentials of your mobile device cloud or AI provider account to configure the connection between Test and the cloud account. See Adding cloud credentials of mobile device clouds on page 1119 and Adding cloud credentials of AI providers on page 1122.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed with the cloud credentials that are added to the team space.
3. Identify the cloud credential by performing any of the following actions:
   - Scroll through the table of cloud credentials to identify the cloud credential with the name you provided.
   - Search for the cloud credential by entering the name of the credential in the **Search** field.
   - Sort the **Provider** column so that the names of the cloud providers are listed alphabetically in either increasing or decreasing order. Identify your cloud provider and look for the name of your credential.

   You identified the cloud credential that you want to use.
4. Click the **Edit cloud credential** icon in the row of the credential that you want to edit.

   The **Update cloud credential** dialog is displayed.
5. Modify the name of the cloud credential in the **Details** tab, and then click **Next**.

   The **Authentication** tab of the **Update cloud credential** dialog is displayed.
6. Modify relevant fields of your subscription account if they have changed, in the **Authentication** tab.

   You can find information about the details that are required for the following account types:
   - Mobile device clouds: Click here on page 1130 to refer to the table.
   - AI providers: Click here on page 1131 to refer to the table.

   The following table lists details for the relevant and required fields for mobile device clouds.

Field	Mobile device cloud					Action
	BitBar Cloud	BrowserStack Cloud	LambdaTest Cloud	pCloudy Cloud	Perfecto Mobile Cloud	
Host	Required	Required	Required	Required	Required	Enter the hostname of the mobile device cloud instance.
API key	Required	Required	Required	Required	Not applicable	Enter the API key of your cloud subscription account to authenticate the connection.

Field	Mobile device cloud					Action
	BitBar Cloud	**BrowserStack Cloud**	**LambdaTest Cloud**	**pCloudy Cloud**	**Perfecto Mobile Cloud**	
Security token	Not applicable	Not applicable	Not applicable	Not applicable	Required	Enter the token used to authenticate the connection.
Project	Required	Required	Required	Not applicable	Not applicable	Enter or select the project to use that is in the mobile device cloud.
Username	Not applicable	Required	Required	Required	Not applicable	Enter the username of your cloud subscription account.

The following table lists details for the relevant and required fields for AI providers.

Attribute	Description	AI Agents			
		OpenAI	**Ollama**	**IBM watsonx**	**Anthropic**
API End Point	The URL of the AI provider.	Optional	Required	Required	Optional
API key	The API key of the AI provider account to authenticate the connection.	Required	Optional	Required	Required
Project Id	A unique identifier assigned to the project that you have created.	Not applicable	Not applicable	Required	Not applicable
Model	The name of the large language model (LLM) that you want to use for test generation.	Not applicable	Optional	Required	Optional

7. Click **Update**.

   The existing cloud credential is updated with the changed details of your account.

Results

You edited an existing cloud credential in your team space and updated it with the changed details of your cloud subscription account.

Related information

Management of cloud credentials on page 1115

Removing authorized users from a cloud credential

When you want to restrict the number of users that are using your cloud credential due to any change in your subscription fee or for other valid reasons, you might want to remove some of the authorized users of the cloud credential in Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Added the cloud credentials of your mobile device cloud or AI provider account to configure the connection between Test and the cloud account. See Adding cloud credentials of mobile device clouds on page 1119 and Adding cloud credentials of AI providers on page 1122.
- Added team space members as authorized users of the cloud credential. See Adding authorized users to a cloud credential on page 1124.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed with the cloud credentials that are added to the team space.
3. Identify the cloud credential by performing any of the following actions:
   - Scroll through the table of cloud credentials to identify the cloud credential with the name you provided.
   - Search for the cloud credential by entering the name of the credential in the **Search** field.
   - Sort the **Provider** column so that the names of the cloud providers are listed alphabetically in either increasing or decreasing order. Identify your cloud provider and look for the name of your credential.

   You identified the cloud credential that you want to use.
4. Perform the following steps to remove authorized users of the cloud credential:
   a. Click the **Add Members** icon in the row of the credential.

      The **Add New Member** dialog is displayed.

   b. Identify the member from the listed members displayed under **Members with access**.

   c. Click in the row of the identified member.

      The selected member is removed from the list displayed under **Members with access**.

   d. Click **Close** to close the dialog.

      The abbreviated name of the authorized user is removed from the display under the **Members** column in the row of the cloud credential.

Results

You as a *Team Space Owner* removed authorized users from the cloud credential.

What to do next

You can inform the team space members that they are removed as authorized users of the cloud credential.

Related information

Management of cloud credentials on page 1115

Deleting an existing cloud credential

When your cloud subscription to the mobile device cloud or AI provider expires and you no longer need the connection from Test , you can delete an existing cloud credential that you configured in your team space.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Added the cloud credentials of your mobile device cloud or AI provider account to configure the connection between Test and the cloud account. See Adding cloud credentials of mobile device clouds on page 1119 and Adding cloud credentials of AI providers on page 1122.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Cloud Credentials** on the navigation pane.

   The **Cloud Credentials** page is displayed with the cloud credentials that are added to the team space.
3. Identify the cloud credential by performing any of the following actions:
   - Scroll through the table of cloud credentials to identify the cloud credential with the name you provided.
   - Search for the cloud credential by entering the name of the credential in the **Search** field.
   - Sort the **Provider** column so that the names of the cloud providers are listed alphabetically in either increasing or decreasing order. Identify your cloud provider and look for the name of your credential.

   You identified the cloud credential that you want to use.
4. Click the **Delete cloud credential** icon in the row of the credential that you want to delete.

   The **Delete cloud credential** dialog is displayed.
5. Click **Delete**.

   The existing cloud credential is deleted and removed from the display on the **Cloud Credentials** page.

Results

You deleted an existing cloud credential that you previously configured, from your team space.

Related information

Management of cloud credentials on page 1115

Test run configurations

You can configure and run tests in Test after you add the test resources to your project in your team space.

Before you configure a test run, you must have completed the following tasks:

- Created a project on Test or you must have been granted access to a project with the *Tester* role assigned. See Managing DevOps Test projects on page 1724.
- Created tests in the desktop clients and committed the test assets and test resources to a remote repository. You must have added the remote repository to the project.
- Read the considerations that you must take into account for certain types of tests. See Prerequisites to running tests on page 1057.

You must be a project *Owner* or a member with the *Tester* role assigned to configure and run a test.

You can find information about how to configure and run the following types of test assets:

Type of Suite or test	Suite or test	More information
AFT Suite	Suite	• Configuring an AFT Suite run on page 1141 • Configuring an AFT Suite to run on a mobile device by using a cloud credential on page 1166
API Suite	Suite	• Configuring an API Suite or an API test run on page 1174
Compound Test containing HTML tests	Suite	• Configuring a run of a Compound Test that contains HTML tests on page 1186

Type of Suite or test	Suite or test	More information
Compound Test containing Web UI tests	Suite	• Configuring a run of a Compound Test that contains Web UI tests on page 1222 • Configuring a run of a Compound Test that contains Web UI tests to run by using variables on page 1235 • Configuring a run of a Compound Test that contains Web UI tests to run by using a cloud credential on page 1249
Compound Test containing mobile tests	Suite	• Configuring a run of a Compound Test that contains mobile tests to run by using variables on page 1197 • Configuring a run of a Compound Test that contains mobile tests to run by using a cloud credential on page 1212
Compound Test containing performance tests	Suite	• Configuring a run of a Compound Test that contains performance tests on page 1260
Compound test containing SAP GUI test		• Configuring a run of an SAP GUI test on page 1334
JMeter test	Suite	• Configuring a run of a JMeter test on page 1377
JUnit test	Suite	• Configuring a run of a JUnit test on page 1386
Postman test	Suite	• Configuring a run of a Postman test on page 1395

Type of Suite or test	Suite or test	More information
Rate Schedule	Suite	• Configuring a run of a Rate Schedule or VU Schedule on page 1404
SAP GUI test	Test	• Configuring a run of an SAP GUI test on page 1334
Selenium test	Suite	• Configuring a run of a Selenium test on page 1417
VU Schedule	Suite	• Configuring a run of a Rate Schedule or VU Schedule on page 1404
API test	Test	• Configuring an API Suite or an API test run on page 1174
Functional test **Note:** The Functional test can contain any of the following individual tests: • Mobile test • Web UI test • Windows test	Test	• Configuring a run of a mobile test to run by using variables on page 1272 • Configuring a run of a mobile test to run on a mobile device by using a cloud credential on page 1286 • Configuring a run of a Web UI test on page 1309 • Configuring a run of a Web UI test to run by using variables on page 1339 • Configuring a run of a Web UI test to run by using a cloud credential on page 1354 • Configuring a run of a Windows test on page 1364
Performance test	Test	• Configuring a run of a Performance test on page 1296
Server Suite asset	Suite	

Type of Suite or test	Suite or test	More information
Server Test asset	Test	• Running server test assets on page 1321 • Running server test assets on an agent on page 1324
Server Suite asset	Suite	Running server suite assets on page 1329

Configuring a Suite or test for a quick run

You can configure any type of Suite or test to be run on Test when you want to quickly ensure that the test runs correctly. You might not want to set the different options for the test nor want to schedule the run.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057 if they apply to the test you want to configure for a run.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:

- Type
- Test Asset Name
- Test Asset Path
- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run, if you want to run a different version other than the latest version.
8. No action is required if you want to initiate the test run immediately after you click **Execute**.
9. Enter a label for the test run that helps you to identify the test on the **Results** page.

After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page.

10. Click **Execute**.
    The test run is initiated.

Results

You have started a test run.

What to do next

You can perform any of the following tasks::

- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Viewing the progress of running test assets on page 1444

Stopping test runs on page 1449

Canceling scheduled test runs on page 1452

Configuring an AFT Suite run

After you added the test resources that you created in the desktop client to the project, you can configure an AFT Suite to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read Test run considerations for AFT Suites or Functional tests on page 1057, if you want to configure a run for an AFT Suite that has the agent location configured in the `AFT XML` file.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.

- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:
   - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

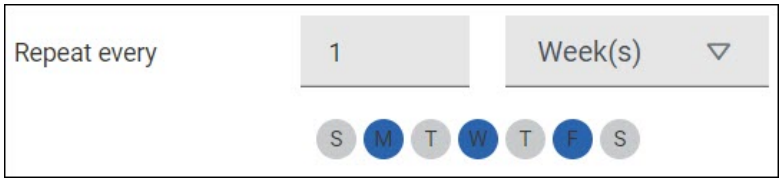
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

If you are running an AFT Suite and the following conditions are true, then you must perform the next step:

- You do not want to run the test on the agent configured in the `AFT XML` file.
- You have not selected the agent from the **Override** column in the **Location** tab.

10. Click **Advanced** to make the following advanced configurations:

    a. Add the following setting in the **Program Arguments** field:

    ```
    -swaplocation <configured_agent_location>:<overriding_agent_location>
    ```

    For example, if the configured agent location is *91.2.352.24* and the remote agent where you want to run the test is *100.35.117.164*, then the entry in the **Program Arguments** field is as follows:

    ```
    -swaplocation 91.2.352.24:100.35.117.164
    ```

    If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project.

Export options	Results in...	Actions
		iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 15 on page 1151.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1152.
15. Follow the instructions if you want to run a test on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the performance agent is displayed.

    Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

    The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the agent.
The agent that is specified in the test assets is available and other agents added to the project.	Performance agents that have capabilities that match with the agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the performance agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

The test run is initiated.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of an AFT Suite.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Unable to run AFT Suites on page 1791

Configuring an AFT Suite to run on a mobile device by using variables

After you added the test resources that you created in the desktop client to the project, you can configure an AFT Suite to run on a mobile device from Test . You must specify the variables for the mobile tests to run on mobile devices that are connected to an agent or a device cloud.

Before you begin

You must have completed the following tasks:

- Read Test run considerations for AFT Suites or Functional tests on page 1057, if you want to configure a run for an AFT Suite that has the agent location configured in the `AFT XML` file.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Created a variable file if you want to import the variables file. The file must contain the details of the UI Test agent or the device cloud to which the mobile device is connected.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

About this task

You can run an AFT Suite, which contains Web UI or mobile tests to run on a mobile device, emulator, or simulator that is connected to an agent or a device cloud. You must create the tests in Test UI, and then add the test asset to your project on Test UI. You can then run the tests on the mobile device, emulator, or simulator that is connected to any of the following agents or clouds:

- Remote UI Test agents. Requires the variables at test run time as specified in Variables for the UI Test agent on page 1155.
- The BitBar Cloud. Requires the variables at test run time as specified in Table 592: Variables for the BitBar Cloud on page 1155.
- The BrowserStack Cloud. Requires the variables at test run time as specified in Table 593: Variables for the BrowserStack Cloud on page 1156.
- The pCloudy Cloud. Requires the variables at test run time as specified in Table 594: Variables for the pCloudy Cloud on page 1156.
- The Perfecto Mobile Cloud. Requires the variables at test run time as specified in Table 595: Variables for the Perfecto Mobile Cloud on page 1157.

You must provide the details of the UI Test agent or the mobile cloud to which the mobile devices are connected as variables in Step 10 on page 1163. You can either enter the variables or use the file in which you entered the variables.

You must refer to the following tables for the variables that are required for a successful run:

Table 591. Variables for the UI Test agent

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the UI Test agent. For example, *Emulator:Pixel_3a_API_33*.
appium.server.host	Specify the hostname or IP address of the UI Test agent.
appium.server.port	Specify the port on the UI Test agent that is configured to communicate with Test.

Table 592. Variables for the BitBar Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BitBar Cloud. For example, *BitBar Cloud:Google Pixel 7 Pro*.
bitbar.apikey	Specify the user token generated for your BitBar Cloud account to authenticate your connection with the BitBar Cloud.
bitbar.host	Specify the hostname of the BitBar Cloud instance.

Table 592. Variables for the BitBar Cloud (continued)

Name of the Variable	Action for the Value field
bitbar.project	Specify the name of the project that contains the recorded test.
bitbar.testrun	Specify a name for the test run that must be displayed in the BitBar Cloud dashboard for the test run.

Table 593. Variables for the BrowserStack Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BrowserStack Cloud. For example, *BrowserStack Cloud:Google Pixel 7 Pro*.
browserstack.host	Specify the URL of the BrowserStack Cloud instance.
browserstack.username	Specify the user name of your BrowserStack Cloud account.
browserstack.apikey	Specify the API key of your BrowserStack Cloud account to authenticate your connection with the BrowserStack Cloud.
browserstack.project	Specify the name of the project that contains the recorded test. **Note:** This variable is optional.

Table 594. Variables for the pCloudy Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the pCloudy Cloud. For example, *pCloudy Cloud:Google Pixel 7 Pro*.
pcloudy.apikey	Specify the API key that is generated for your pCloudy Cloud account to authenticate your connection with the pCloudy Cloud.

Table 594. Variables for the pCloudy Cloud (continued)

Name of the Variable	Action for the Value field
pcloudy.username	Specify the username with which you can access the pCloudy Cloud instance.
pcloudy.host	Specify the hostname of the pCloudy Cloud instance.

Table 595. Variables for the Perfecto Mobile Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the Perfecto Mobile Cloud. For example, *Perfecto Mobile Cloud:Google Pixel 7 Pro*.
perfecto.securitytoken	Specify the user token generated for your Perfecto Mobile Cloud account to authenticate your connection with the Perfecto Mobile Cloud.
perfecto.host	Specify the hostname of the Perfecto Mobile Cloud instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.

5. Identify the test asset that you want to run by performing any of the following actions:
    - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

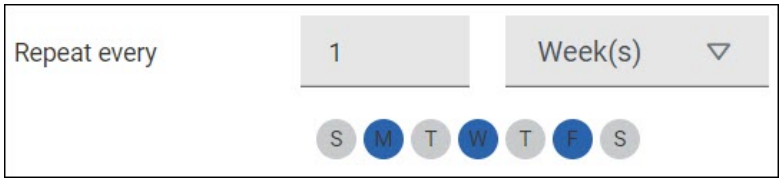
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps to provide the variables that specify the device cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

11. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.

Export options	Results in...	Actions
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

12. Perform any of the following actions that depend on whether you are using a remote performance agent or a mobile device cloud:
    - Go to Step 13 on page 1164, if you are using an agent.
    - Go to Step 14 on page 1165, if you are using a mobile device cloud.
13. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:
    a. Click the **LOCATION** tab, if it is not already open.

       The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

       Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

       The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

14. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of an AFT Suite to run on a device, emulator, or simulator that is connected to an agent or mobile device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring an AFT Suite to run on a mobile device by using a cloud credential

After you added the test resources that you created in the desktop client to the project, you can configure an AFT Suite to be run on a mobile device by using a cloud credential from Test .

Before you begin

You must have completed the following tasks:

- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:

a. Click **New filter**.
b. Enter a name for the filter.
c. Select an operator, and then add a rule or a group of rules.
d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

   You can select a parameter from the following list:
   - Type
   - Test Asset Name
   - Test Asset Path
   - Last Result
   - Next Run
   - Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

   The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

  Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

  a. Click the **Filter** icon in the **ADVANCED** tab.

     The filter queries that you created and saved are displayed.

  b. Click the filter that you want to apply.

     The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

   Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

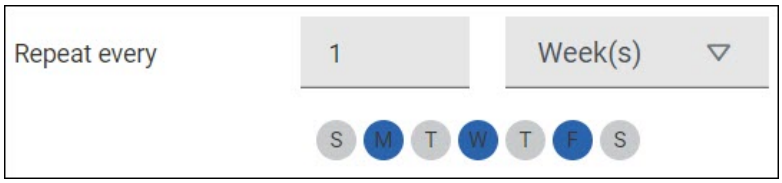
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps in the **ENVIRONMENT** tab:

    a. Select the cloud credential of the mobile device cloud from the **Device cloud credentials** list.

       The list displays the cloud credentials that are configured in the team space. The name of the cloud credential is appended to the name of the mobile device cloud in the list. For example, if the mobile device cloud provider is *BitBar* and the name of the credential is *BB_Account1*, then the name displayed in the list is *Bit Bar - BB_Account1*.

Note: Access to credentials for mobile device cloud services depends on the status of the provider. If you disable a provider on the **Cloud Credentials** page, you cannot access that device cloud provider from the **Device cloud credentials** list.

The devices that are available for testing on the selected mobile device cloud are not displayed because the devices are already specified in the AFT Suite.

Note: If you are connected to either the BitBar Cloud or BrowserStack Cloud, the **Projects** field is displayed.

b. Select the project on the mobile device cloud that you want to use or create a project in the **Projects** field. The project overrides the project that is configured in the AFT Suite.

11. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of an AFT Suite to run on a mobile device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring an API Suite or an API test run

After you added the test resources that you created in the desktop client to the project, you can configure an API Suite or an API test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the following tasks if you are running API Suites or tests that use a transport and the transport requires third-party application `Jar` files for a successful run:
  - Read Test run considerations for API Suites or API tests on page 1059.
  - Copied the third-party application `Jar` files to the third-party application folder on the computer where Test is installed, if you are running the API Suite in Kubernetes. See Copying third-party application Jars to Kubernetes on page 736.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.

Tab	Description
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.

d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

You can select a parameter from the following list:

- Type
- Test Asset Name
- Test Asset Path
- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

◦ Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

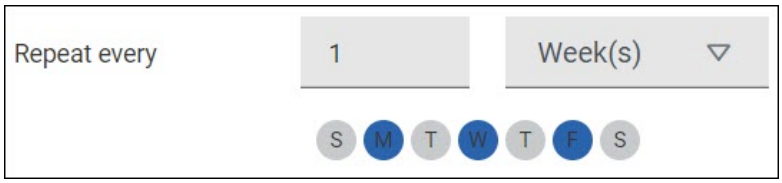
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       Perform the following actions, if applicable for your test:

       ▪ Configuring test log reporting.

         If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

- Overriding results database that is configured in the API Suite project.

Note: You can use any of the databases that are specified in the system requirements as supported databases in Test Integrations and APIs.

Enter the following arguments in the **Program Arguments** field, if you want to override the **Results Database**:

Program Arguments	Description
-resultDatabaseURL *<database_URL>*	Specifies the URL of the results database that overrides the results database that is configured for the project in Test Integrations and APIs.
-resultDatabaseDriver *<driver_name>*	Specifies the driver of the results database that overrides the results database that is configured for the project in Test Integrations and APIs. For example, the driver for a MySQL database is *com.mysql.cj.jdbc.Driver*.
-resultDatabaseUsername *<user_name>*	Specifies the username that is authorized to access the results database that overrides the results database that is configured for the project in Test Integrations and APIs.
-resultDatabasePassword *<encrypted_password>*	Specifies the password that is to be used to authenticate the user to access the results database that overrides the results database that is configured for the project in Test Integrations and APIs. **Note:** You can encrypt the password by using the **EncryptPassword** application that is available in the installation directory of Test Integrations and APIs.

For example, if you want to override the results database that is configured in the Test Integrations and APIs project with a different results database that has the following details:

- MySQL database as the results database with the URL as *jdbc:mysql://localhost:3306/myresultsdb*
- *com.mysql.cj.jdbc.Driver* as the driver for the MySQL database
- *tester* as the username to access the MySQL database
- #com.ghc.*1!9a8521137E0A12994ACF4434E315A2B9* as the encrypted password to authenticate the user name *tester*

The entry in the **Program Arguments** field must be as follows:

```
-resultDatabaseURL  jdbc:mysql://localhost:3306/myresultsdb
-resultDatabaseDriver com.mysql.cj.jdbc.Driver
```

```
-resultDatabaseUsername tester
-resultDatabasePassword #com.ghc.1!9a8521137E0A12994ACF4434E315A2B9
```

Note: You must separate entries in the field with white spaces or with a new-line character.

11. Follow the instructions if the API Suite has an environment or secrets configured:
    a. Click the **ENVIRONMENT** tab, if it is not already open.
    b. Select the API test environment from the list if there are multiple environments configured in the test asset.
    c. Select the classification that contains the secrets to be used for the test run.

Notes:

- The test asset that was created in the desktop client and added to the Git repository must have the environments defined as part of the API test project.
- If the test asset contained secrets, then you must create those secrets in the classifications in the project.

The default value for the environment is the environment configured in the test asset. The default value for secrets is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

    Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

13. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

a. Click the **VARIABLES** tab, if it is not already open.

b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

The default value for the variables is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

14. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

a. Click the **RESULTS** tab.

b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

15. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

    Kubernetes clusters that are registered with a team space.

    Notes:

    - If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
    - If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

    - Select the **Default Cluster** when no remote clusters are available in your project.
    - Select the remote cluster from the list when a remote cluster is available in your project.
    - Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

    c. Click **Execute**.

    The test run is initiated.

Results

You have started or scheduled a test run of an API Suite.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Compound Test that contains HTML tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains HTML tests to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- When you want to run a Compound Test, you must have authored a Compound Test in HCL DevOps Test UI (Test UI) and must have completed the following tasks when you created the Compound Test:
  - Created a Compound Test under a Web UI project in your workspace.
  - Added a Functional HTML Test under the Compound Test.
  - Used the **startBrowser(Firefox,<URL>)** attribute with the browser value set to *Firefox* in the Functional test script.

    Restriction: Functional test scripts that contain the **startApp()** or **callScript()** attributes are treated as non-HTML scripts and are not supported to be used in test scripts in the HTML tests.

  - Linked the dataset that is used in the Functional HTML Test with the Web UI Compound Test only if you want to override this dataset. You must have created the test resources in HCL DevOps Test UI, before committing the test resources to the server project repository. You must have also created the dataset that you want to use to override the dataset linked to the test.
  - Ran the Compound Test that contains the HTML tests on Test UI.
  - Committed the projects that include the Web UI project (with the Compound Test) and the Functional test (with the HTML tests) to the remote repository.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.

- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

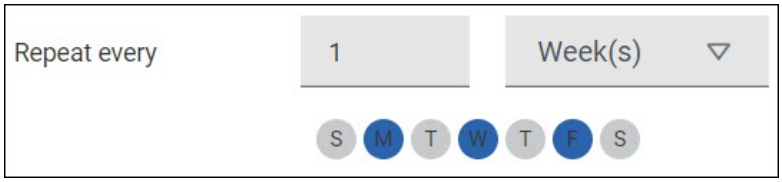
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions in this step if the test requires a variable that must be passed to the test at the test run time.

    You must configure the supported browser by using a variable if the test has a browser configured, which is different from the one that is supported by Test.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.

        i. Enter `RTW_WebUI_Browser_Selection` as the name.
        ii. Enter `Firefox` as the value, if you want to use Firefox as the browser and override the browser that is specified in the HTML test.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

    Kubernetes clusters that are registered with a team space.

    Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains Functional HTML tests.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Compound Test that contains mobile tests to run by using variables

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains mobile tests to run by using variables from Test . You must specify the variables for the mobile tests to run on mobile devices that are connected to an agent or a device cloud.

Before you begin

You must have completed the following tasks:

- Created a Compound Test that contains the mobile tests to run on a mobile device in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Created a variable file if you want to import the variables file. The file must contain the details of the UI Test agent or the device cloud to which the mobile device is connected.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run a Compound Test, which contains mobile tests to run on a mobile device, emulator, or simulator that is connected to an agent or a device cloud. You must create the tests in Test UI, and then add the test asset to your project on Test. You can then run the mobile tests on the mobile devices that are connected to any of the following agents or clouds:

- Remote UI Test agents. Requires the variables at test run time as specified in Variables for the UI Test agent on page 1199.
- The BitBar Cloud. Requires the variables at test run time as specified in Variables for the BitBar Cloud on page 1199.
- The BrowserStack Cloud. Requires the variables at test run time as specified in Variables for the BrowserStack Cloud on page 1199.
- The pCloudy Cloud. Requires the variables at test run time as specified in Variables for the pCloudy Cloud on page 1200.
- The Perfecto Mobile Cloud. Requires the variables at test run time as specified in Variables for the Perfecto Mobile Cloud on page 1200.

You must provide the details of the UI Test agent or the mobile cloud to which the mobile devices are connected as variables in Step 12 on page 1208. You can either enter the variables or use the file in which you entered the variables.

You must refer to the following tables for the variables that are required for a successful run:

Table 596. Variables for the UI Test agent

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the UI Test agent. For example, *Emulator:Pixel_3a_API_33*.
appium.server.host	Specify the hostname or IP address of the UI Test agent.
appium.server.port	Specify the port on the UI Test agent that is configured to communicate with Test.

Table 597. Variables for the BitBar Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BitBar Cloud. For example, *BitBar Cloud:Google Pixel 7 Pro*.
bitbar.apikey	Specify the user token generated for your BitBar Cloud account to authenticate your connection with the BitBar Cloud.
bitbar.host	Specify the hostname of the BitBar Cloud instance.
bitbar.project	Specify the name of the project that contains the recorded test.
bitbar.testrun	Specify a name for the test run that must be displayed in the BitBar Cloud dashboard for the test run.

Table 598. Variables for the BrowserStack Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BrowserStack Cloud. For example, *BrowserStack Cloud:Google Pixel 7 Pro*.

Table 598. Variables for the BrowserStack Cloud (continued)

Name of the Variable	Action for the Value field
browserstack.host	Specify the URL of the BrowserStack Cloud instance.
browserstack.username	Specify the user name of your BrowserStack Cloud account.
browserstack.apikey	Specify the API key of your BrowserStack Cloud account to authenticate your connection with the BrowserStack Cloud.
browserstack.project	Specify the name of the project that contains the recorded test. **Note:** This variable is optional.

Table 599. Variables for the pCloudy Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the pCloudy Cloud. For example, *pCloudy Cloud:Google Pixel 7 Pro*.
pcloudy.apikey	Specify the API key that is generated for your pCloudy Cloud account to authenticate your connection with the pCloudy Cloud.
pcloudy.username	Specify the username with which you can access the pCloudy Cloud instance.
pcloudy.host	Specify the hostname of the pCloudy Cloud instance.

Table 600. Variables for the Perfecto Mobile Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the Perfecto Mobile Cloud. For example, *Perfecto Mobile Cloud:Google Pixel 7 Pro*.

Table 600. Variables for the Perfecto Mobile Cloud (continued)

Name of the Variable	Action for the Value field
perfecto.securitytoken	Specify the user token generated for your Perfecto Mobile Cloud account to authenticate your connection with the Perfecto Mobile Cloud.
perfecto.host	Specify the hostname of the Perfecto Mobile Cloud instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.

- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
    a. Click **New filter**.
    b. Enter a name for the filter.
    c. Select an operator, and then add a rule or a group of rules.
    d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

    You can select a parameter from the following list:
        - Type
        - Test Asset Name
        - Test Asset Path
        - Last Result
        - Next Run
        - Components

    e. Save the filter query to save and apply the filter query to filter the assets based on the query.

    The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

    Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

    a. Click the **Filter** icon in the **ADVANCED** tab.

    The filter queries that you created and saved are displayed.

    b. Click the filter that you want to apply.

    The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

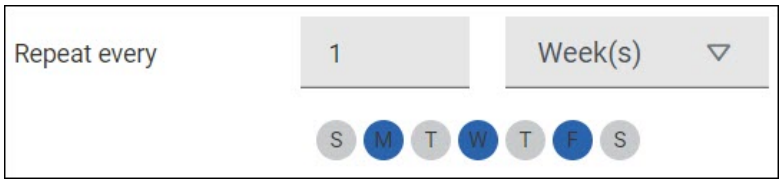
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter the following program argument that must be passed to the test run at run time in the **Program Arguments** field:

       Enter `-history testlog` in the **Program Arguments** field.

Important: If the `-history testlog` is not entered in the **Program Arguments** field, the unified report that is generated displays the test steps incorrectly.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Perform the following steps to provide the variables that specify the UI Test agent or cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Perform any of the following actions that depend on whether you are using a remote performance agent or a mobile device cloud:
    - Go to Step 15 on page 1210, if you are using an agent.
    - Go to Step 16 on page 1211, if you are using a mobile device cloud.
15. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:
    a. Click the **LOCATION** tab, if it is not already open.

       The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

       Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

       The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent.

If...	Then...	Action
performance agents are added to the project.		You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

16. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains mobile tests to run on a mobile device that is connected to an agent or a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Compound Test that contains mobile tests to run by using a cloud credential

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains mobile tests to run by using a cloud credential from Test .

Before you begin

You must have completed the following tasks:

- Created a Compound Test that contains the mobile tests in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

About this task

You can run a Compound Test, which contains mobile tests to run on mobile devices that are connected to a device cloud. You must create the tests in Test UI, and then add the test asset to your project on Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:

a. Click **New filter**.
b. Enter a name for the filter.
c. Select an operator, and then add a rule or a group of rules.
d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

   You can select a parameter from the following list:
   - Type
   - Test Asset Name
   - Test Asset Path
   - Last Result
   - Next Run
   - Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

   The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

  Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

  a. Click the **Filter** icon in the **ADVANCED** tab.

     The filter queries that you created and saved are displayed.

  b. Click the filter that you want to apply.

     The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

   Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

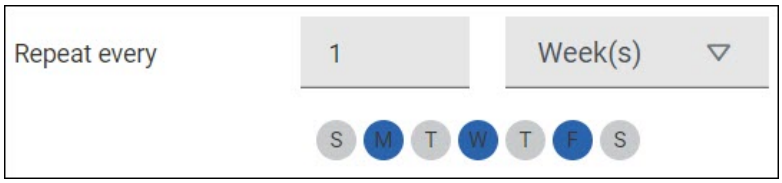
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps in the **ENVIRONMENT** tab:

    a. Select the cloud credential of the mobile device cloud from the **Device cloud credentials** list.

       The list displays the cloud credentials that are configured in the team space. The name of the cloud credential is appended to the name of the mobile device cloud in the list. For example, if the mobile device cloud provider is *BitBar* and the name of the credential is *BB_Account1*, then the name displayed in the list is *Bit Bar - BB_Account1*.

Note: Access to credentials for mobile device cloud services depends on the status of the provider. If you disable a provider on the **Cloud Credentials** page, you cannot access that device cloud provider from the **Device cloud credentials** list.

If the connection to the mobile device cloud is successful, the **Mobile devices** field is displayed.

b. Select the devices on which you want to run the tests from the list of devices that are displayed in the **Mobile devices** list.

Note: If you are connected to either the BitBar Cloud or BrowserStack Cloud, the **Projects** field is displayed.

c. Select the project on the mobile device cloud that you want to use or create a project in the **Projects** field. The project overrides the project that is configured in the test asset.

11. Click **Advanced** to make the following advanced configurations:

a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

For example, you can set a maximum Java heap size.

b. Enter the following program argument that must be passed to the test run at run time in the **Program Arguments** field:

Enter `-history testlog` in the **Program Arguments** field.

Important: If the `-history testlog` is not entered in the **Program Arguments** field, the unified report that is generated displays the test steps incorrectly.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data.

Option	Description
	Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains mobile tests to run on a mobile device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring a run of a Compound Test that contains Web UI tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains Web UI tests to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Created a Compound Test that contains Web UI tests in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

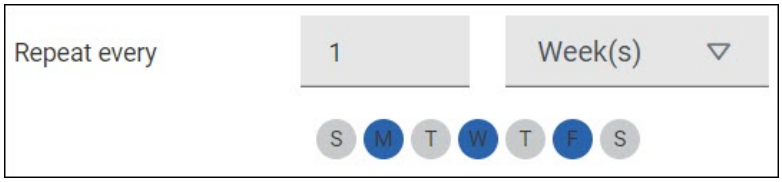
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

    Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

    The default value for each of the fields for the advanced settings is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions in this step if the test requires a variable that must be passed to the test at the test run time.

    You must configure the supported browser by using a variable if the test has a browser configured, which is different from the one that is supported by Test.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.

        i. Enter `RTW_WebUI_Browser_Selection` as the name.
        ii. Enter `Firefox` as the value, if you want to use Firefox as the browser and override the browser that is specified in the Web UI test.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 15 on page 1232.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1234.
15. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	**Then...**	**Action**
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

If...	Then...	Action
	You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	

c. Click **Execute**.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains Web UI tests.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Compound Test that contains Web UI tests to run by using variables

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains Web UI tests to run by using variables from Test . You must specify the variables for the Web UI tests to run on mobile devices that are connected to an agent or a device cloud.

Before you begin

You must have completed the following tasks:

- Created a Compound Test that contains Web UI tests to run on a mobile device in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Created a variable file if you want to import the variables file. The file must contain the details of the UI Test agent or the device cloud to which the mobile device is connected.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run the Compound Test that contains Web UI tests on a mobile device, emulator, or simulator that is connected to an agent or a device cloud. You must create the tests in Test UI, and then add the test asset to your project on Test. You can then run the Compound Test that contains Web UI tests on a mobile device that is connected to any of the following agents or clouds:

- Remote UI Test agents. Requires the variables at test run time as specified in Variables for the UI Test agent on page 1236.
- The BitBar Cloud. Requires the variables at test run time as specified in Variables for the BitBar Cloud on page 1237.
- The BrowserStack Cloud. Requires the variables at test run time as specified in Variables for the BrowserStack Cloud on page 1237.
- The pCloudy Cloud. Requires the variables at test run time as specified in Variables for the pCloudy Cloud on page 1238.
- The Perfecto Mobile Cloud. Requires the variables at test run time as specified in Variables for the Perfecto Mobile Cloud on page 1238.

You must provide the details of the UI Test agent or the mobile cloud to which the mobile devices are connected as variables in Step 12 on page 1246. You can either enter the variables or use the file in which you entered the variables.

You must refer to the following tables for the variables that are required for a successful run:

Table 601. Variables for the UI Test agent

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use on the UI Test agent. For example, *Chrome(Emulator:Pixel_3a_API_33)*.
appium.server.host	Specify the hostname or IP address of the UI Test agent.
appium.server.port	Specify the port on the UI Test agent that is configured to communicate with Test.

Table 602. Variables for the BitBar Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the BitBar Cloud. For example, *Chrome(BitBar Cloud:Google Pixel 7 Pro)*.
bitbar.apikey	Specify the user token generated for your BitBar Cloud account to authenticate your connection with the BitBar Cloud.
bitbar.host	Specify the hostname of the BitBar Cloud instance.
bitbar.project	Specify the name of the project that contains the recorded test.
bitbar.testrun	Specify a name for the test run that must be displayed in the BitBar Cloud dashboard for the test run.

Table 603. Variables for the BrowserStack Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the BrowserStack Cloud. For example, *Chrome(BrowserStack Cloud:Google Pixel 7 Pro)*.
browserstack.host	Specify the URL of the BrowserStack Cloud instance.
browserstack.username	Specify the user name of your BrowserStack Cloud account.
browserstack.apikey	Specify the API key of your BrowserStack Cloud account to authenticate your connection with the BrowserStack Cloud.
browserstack.project	Specify the name of the project that contains the recorded test.

Table 603. Variables for the BrowserStack Cloud (continued)

Name of the Variable	Action for the Value field
	Note: This variable is optional.

Table 604. Variables for the pCloudy Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the pCloudy Cloud. For example, *Chrome(pCloudy Cloud:Google Pixel 7 Pro)*.
pcloudy.apikey	Specify the API key that is generated for your pCloudy Cloud account to authenticate your connection with the pCloudy Cloud.
pcloudy.username	Specify the username with which you can access the pCloudy Cloud instance.
pcloudy.host	Specify the hostname of the pCloudy Cloud instance.

Table 605. Variables for the Perfecto Mobile Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the Perfecto Mobile Cloud. For example, *Chrome(Perfecto Mobile Cloud:Google Pixel 7 Pro)*.
perfecto.securitytoken	Specify the user token generated for your Perfecto Mobile Cloud account to authenticate your connection with the Perfecto Mobile Cloud.
perfecto.host	Specify the hostname of the Perfecto Mobile Cloud instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.

3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path

- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

◦ Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

◦ If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

◦ If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

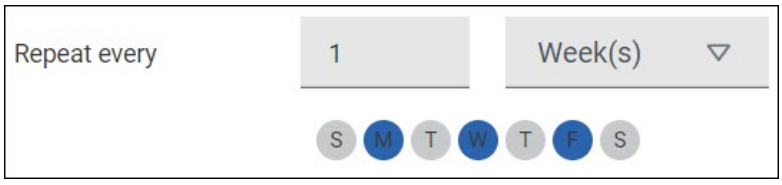
The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.
   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options:

Schedule	Actions
	▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon . You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

    Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

    - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

      Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

    - Select the dataset from the list.

      Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

    - Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

      Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added

datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Perform the following steps to provide the variables that specify the UI Test agent or cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

a. Click the **RESULTS** tab.

b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Perform any of the following actions that depend on whether you are using a remote performance agent or a mobile device cloud:
    - Go to Step 15 on page 1247, if you are using an agent.
    - Go to Step 16 on page 1249, if you are using a mobile device cloud.
15. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

a. Click the **LOCATION** tab, if it is not already open.

The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	**Then...**	**Action**
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

If...	Then...	Action
	match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	

c. Click **Execute**.

16. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains Web UI tests to run on a mobile device that is connected to an agent or a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Compound Test that contains Web UI tests to run by using a cloud credential

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains Web UI tests to run on devices by using a cloud credential from Test .

Before you begin

You must have completed the following tasks:

- Created a Compound Test that contains the Web UI tests in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

About this task

You can run a Compound Test, which contains Web UI tests on a device cloud. You must create the tests in Test UI, and then add them to your project on Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:
    - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

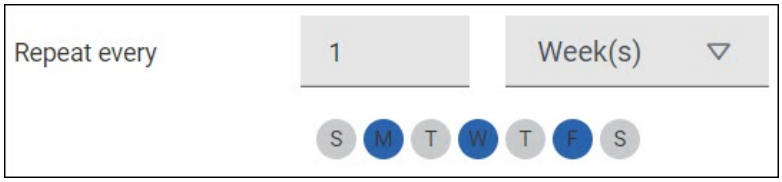
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps in the **ENVIRONMENT** tab:

    a. Select the cloud credential of the mobile device cloud from the **Device cloud credentials** list.

    The list displays the cloud credentials that are configured in the team space. The name of the cloud credential is appended to the name of the mobile device cloud in the list. For example, if the mobile device cloud provider is *BitBar* and the name of the credential is *BB_Account1*, then the name displayed in the list is *Bit Bar - BB_Account1*.

    Note: Access to credentials for mobile device cloud services depends on the status of the provider. If you disable a provider on the **Cloud Credentials** page, you cannot access that device cloud provider from the **Device cloud credentials** list.

    If the connection to the mobile device cloud is successful, the **Mobile devices** field is displayed.

    b. Select the devices on which you want to run the tests from the list of devices that are displayed in the **Mobile devices** list.

    Note: If you are connected to either the BitBar Cloud or BrowserStack Cloud, the **Projects** field is displayed.

    c. Select the project on the mobile device cloud that you want to use or create a project in the **Projects** field. The project overrides the project that is configured in the test asset.

11. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

    For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

    If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains Web UI tests to run on a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.

- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring a run of a Compound Test that contains performance tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains performance tests to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Created performance tests under a Compound Test in HCL DevOps Test Performance (Test Performance).
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **SUITES** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:
    - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path

- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

◦ Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

◦ If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

◦ If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, AFT XML files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

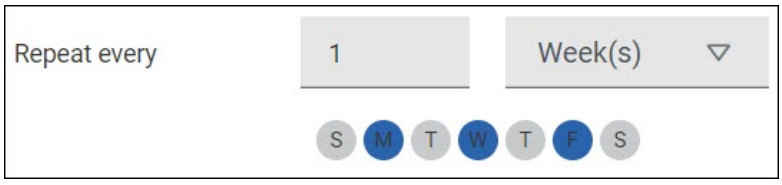
The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.
   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options:

Schedule	Actions
	▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

    Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

    - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

      Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

    - Select the dataset from the list.

      Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

    - Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

      Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added

datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

a. Click the **RESULTS** tab.

b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 15 on page 1269.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1271.
15. Follow the instructions if you want to run a test on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the performance agent is displayed.

Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the agent.
The agent that is specified in the test assets is available and other agents added to the project.	Performance agents that have capabilities that match with the agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

If...	Then...	Action
	hover over the performance agent in the **Override** column.	

c. Click **Execute**.

The test run is initiated.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Compound Test that contains performance tests.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a mobile test to run by using variables

After you added the test resources that you created in the desktop client to the project, you can configure a mobile test to run by using variables on a mobile device from Test . You must specify the variables for the mobile test to run on mobile devices that are connected to an agent or a device cloud.

Before you begin

You must have completed the following tasks:

- Created a mobile test in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Created a variable file if you want to import the variables file. The file must contain the details of the UI Test agent or the device cloud to which the mobile device is connected.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run the mobile test after you create the test in Test UI, and then add them to your project on Test. You can then run the mobile test on the mobile devices that are connected to any of the following agents or clouds:

- Remote UI Test agents. Requires the variables at test run time as specified in Variables for the UI Test agent on page 1273.
- The BitBar Cloud. Requires the variables at test run time as specified in Variables for the BitBar Cloud on page 1273.
- The BrowserStack Cloud. Requires the variables at test run time as specified in Variables for the BrowserStack Cloud on page 1274.
- The pCloudy Cloud. Requires the variables at test run time as specified in Variables for the pCloudy Cloud on page 1274.
- The Perfecto Mobile Cloud. Requires the variables at test run time as specified in Variables for the Perfecto Mobile Cloud on page 1275.

You must provide the details of the UI Test agent or the mobile cloud to which the mobile devices are connected as variables in Step 12 on page 1283. You can either enter the variables or use the file in which you entered the variables.

You must refer to the following tables for the variables that are required for a successful run:

Table 606. Variables for the UI Test agent

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the UI Test agent. For example, *Emulator:Pixel_3a_API_33*.
appium.server.host	Specify the hostname or IP address of the UI Test agent.
appium.server.port	Specify the port on the UI Test agent that is configured to communicate with Test.

Table 607. Variables for the BitBar Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BitBar Cloud. For example, *BitBar Cloud:Google Pixel 7 Pro*.
bitbar.apikey	Specify the user token generated for your BitBar Cloud account to authenticate your connection with the BitBar Cloud.

Table 607. Variables for the BitBar Cloud (continued)

Name of the Variable	Action for the Value field
bitbar.host	Specify the hostname of the BitBar Cloud instance.
bitbar.project	Specify the name of the project that contains the recorded test.
bitbar.testrun	Specify a name for the test run that must be displayed in the BitBar Cloud dashboard for the test run.

Table 608. Variables for the BrowserStack Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the BrowserStack Cloud. For example, *BrowserStack Cloud:Google Pixel 7 Pro*.
browserstack.host	Specify the URL of the BrowserStack Cloud instance.
browserstack.username	Specify the user name of your BrowserStack Cloud account.
browserstack.apikey	Specify the API key of your BrowserStack Cloud account to authenticate your connection with the BrowserStack Cloud.
browserstack.project	Specify the name of the project that contains the recorded test. **Note:** This variable is optional.

Table 609. Variables for the pCloudy Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the pCloudy Cloud. For example, *pCloudy Cloud:Google Pixel 7 Pro*.

Table 609. Variables for the pCloudy Cloud (continued)

Name of the Variable	Action for the Value field
pcloudy.apikey	Specify the API key that is generated for your pCloudy Cloud account to authenticate your connection with the pCloudy Cloud.
pcloudy.username	Specify the username with which you can access the pCloudy Cloud instance.
pcloudy.host	Specify the hostname of the pCloudy Cloud instance.

Table 610. Variables for the Perfecto Mobile Cloud

Name of the Variable	Action for the Value field
Mobile_Device_Selection	Specify the name of the mobile device that is connected to the Perfecto Mobile Cloud. For example, *Perfecto Mobile Cloud:Google Pixel 7 Pro*.
perfecto.securitytoken	Specify the user token generated for your Perfecto Mobile Cloud account to authenticate your connection with the Perfecto Mobile Cloud.
perfecto.host	Specify the hostname of the Perfecto Mobile Cloud instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.

Tab	Description
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **TESTS** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:
    - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

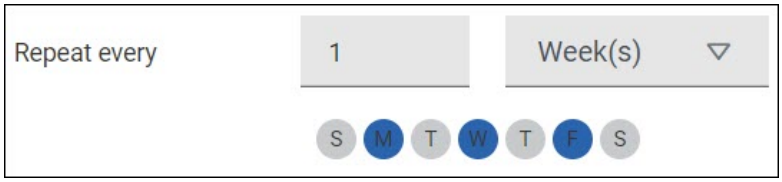
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Perform the following steps to provide the variables that specify the UI Test agent or cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project.

Export options	Results in...	Actions
		iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Perform any of the following actions that depend on whether you are using a remote performance agent or a mobile device cloud:
    - Go to Step 15 on page 1284, if you are using an agent.
    - Go to Step 16 on page 1285, if you are using a mobile device cloud.
15. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:
    a. Click the **LOCATION** tab, if it is not already open.

       The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

       Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

       The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

16. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a mobile test to be run on a mobile device that is connected to an agent or device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a mobile test to run on a mobile device by using a cloud credential

After you added the test resources that you created in the desktop client to the project, you can configure a mobile test to run on a mobile device cloud by using a cloud credential from Test .

Before you begin

You must have completed the following tasks:

- Created a mobile test in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

About this task

You can run the mobile test on a mobile device cloud. You must create the test in Test UI, and then add it to your project on Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **TESTS** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

     Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

     Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
JMeter	JMeter Test	✓		✓
Ju	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
Se	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.

- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
    a. Click **New filter**.
    b. Enter a name for the filter.
    c. Select an operator, and then add a rule or a group of rules.
    d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

    You can select a parameter from the following list:
        - Type
        - Test Asset Name
        - Test Asset Path
        - Last Result
        - Next Run
        - Components

    e. Save the filter query to save and apply the filter query to filter the assets based on the query.

    The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

    Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

    a. Click the **Filter** icon in the **ADVANCED** tab.

    The filter queries that you created and saved are displayed.

    b. Click the filter that you want to apply.

    The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

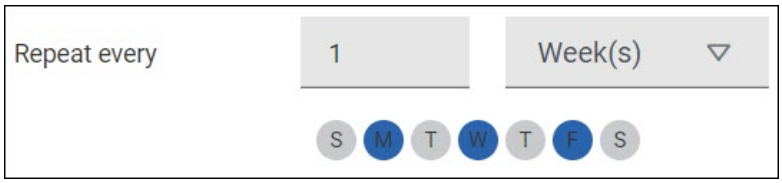
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps in the **ENVIRONMENT** tab:

    a. Select the cloud credential of the mobile device cloud from the **Device cloud credentials** list.

       The list displays the cloud credentials that are configured in the team space. The name of the cloud credential is appended to the name of the mobile device cloud in the list. For example, if the mobile device cloud provider is *BitBar* and the name of the credential is *BB_Account1*, then the name displayed in the list is *Bit Bar - BB_Account1*.

Note: Access to credentials for mobile device cloud services depends on the status of the provider. If you disable a provider on the **Cloud Credentials** page, you cannot access that device cloud provider from the **Device cloud credentials** list.

If the connection to the mobile device cloud is successful, the **Mobile devices** field is displayed.

b. Select the devices on which you want to run the tests from the list of devices that are displayed in the **Mobile devices** list.

Note: If you are connected to either the BitBar Cloud or BrowserStack Cloud, the **Projects** field is displayed.

c. Select the project on the mobile device cloud that you want to use or create a project in the **Projects** field. The project overrides the project that is configured in the test asset.

11. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

    For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

    If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

    For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.

Option	Description
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a mobile test to run on a mobile device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring a run of a Performance test

After you added the test resources that you created in the desktop client to the project, you can configure a Performance test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Created a Performance test in HCL DevOps Test Performance (Test Performance).
- Read Test run considerations for using Machine learning capabilities on page 1064, if you want to configure a run for either a Performance Test or a Schedule that must be analyzed by using Machine Learning capabilities in Test.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **TESTS** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

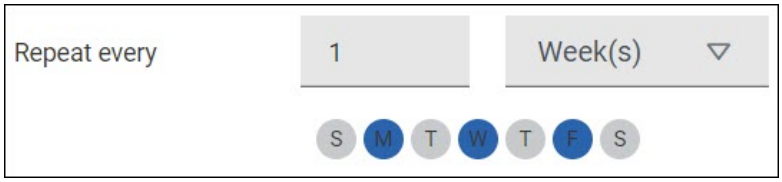
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

    Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

    The default value for each of the fields for the advanced settings is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    Note: You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project.

Export options	Results in...	Actions
		iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 15 on page 1306.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1307.
15. Follow the instructions if you want to run a test on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the performance agent is displayed.

    Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

    The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the agent.
The agent that is specified in the test assets is available and other agents added to the project.	Performance agents that have capabilities that match with the agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the performance agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

The test run is initiated.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Performance test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Web UI test

After you added the test resources that you created in the desktop client to the project, you can configure a Web UI test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Created a Web UI test in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.

- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
    a. Click **New filter**.
    b. Enter a name for the filter.
    c. Select an operator, and then add a rule or a group of rules.
    d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

    You can select a parameter from the following list:
        - Type
        - Test Asset Name
        - Test Asset Path
        - Last Result
        - Next Run
        - Components

    e. Save the filter query to save and apply the filter query to filter the assets based on the query.

    The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

    Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

    a. Click the **Filter** icon in the **ADVANCED** tab.

    The filter queries that you created and saved are displayed.

    b. Click the filter that you want to apply.

    The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

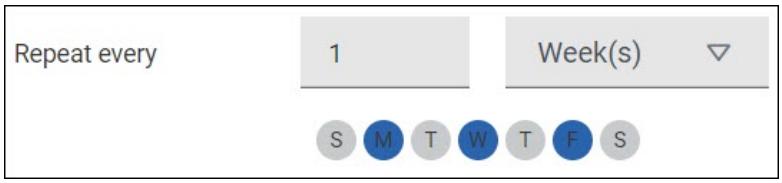
- No action is required if you want to initiate the test run immediately after you click **Execute**.

  Important: Click **Execute** only after you have configured the other settings in this dialog.

  Note: The default time for scheduling a run is **Now**.

- Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Follow the instructions in this step if the test requires a variable that must be passed to the test at the test run time.

    You must configure the supported browser by using a variable if the test has a browser configured, which is different from the one that is supported by Test.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    i. Enter `RTW_WebUI_Browser_Selection` as the name.
    ii. Enter `Firefox` as the value, if you want to use Firefox as the browser and override the browser that is specified in the Web UI test.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

The default value for the variables is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

13. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 14 on page 1318.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1320.
14. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

    Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

    The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.

If...	Then...	Action
performance agents are added to the project.		
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

15. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

    Kubernetes clusters that are registered with a team space.

    Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Web UI test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Running server test assets

You can run tests that are listed as server test assets created in Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created a Web UI test in Test. See Recording a Web UI test on page 771.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select one of the following tabs based on the type of test assets that you want to run:
   - SUITES
   - TESTS
   - ADVANCED

   The test assets in the selected tab are displayed.
5. Identify the Server Test asset that you want to run by scrolling through the list, searching for the test, or applying a filter.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.
   The **Execute test asset** dialog is displayed.
7. Select the version of the test resources that you want to run in the **Version** field.
8. Click **Schedule** and select the time if you want to schedule the test run. Else, skip the step.
9. Enter a label for the test run that helps you to identify the test on the **Results** page.
10. Follow the instructions to select a browser:

    a. Click the **ENVIRONMENT** tab, if it is not already open.

    b. Select an environment from the list that is available for the test run from the **Environment** drop-down list.
    The variables and profile overrides in the selected environment override the values defined in the test when you run a test or suite.

    c. Select a browser or device profile for the test run from the **Browser** drop-down list.
    The browser that is used to record the test is selected as the default browser and the browser specified here overrides the default browser and the browser defined in the profile override.

11. **Optional:** Click the **DATA SOURCES** and select a dataset tab if you are running a test asset that contains datasets.
12. **Optional:** Click the **VARIABLES** tab, and then add or upload variables if the test requires a variable that must be passed to the test at the test run time.
13. **Optional:** Click the **LOCATION** tab, and then select the agent that you have added in the **Override** field.
    After selecting the agent, if the specified browser is not available to the agent, Test displays a warning indicator in the **Environment** tab.
14. Click the **RESULTS** tab and select an option if you want to export the test results to a Jira issue in your Xray project in Jira.
    You must have set up Jira as a change management system and configured the Xray integration to export the results.
15. Click **Advanced** to enter any of the following values that must be passed to the test run if they are applicable for the test:
    - **JVM Arguments**
    - **Program Arguments**
    - **Environment Variables**
16. Click **Execute**.
    The test run is initiated.

Results

You have run a server test asset that is created in Test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Running server test assets on an agent

When you have server test assets that you want to run on a Linux, Mac, or Windows system by using an agent, you can register the agents and run the tests.

Before you begin

You must have completed the following tasks:

- Registered the Linux, Mac, or Windows system using an agent for running tests. See Installing and registering DevOps Test Runtime agents on page 1087.
- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Created a Web UI test in Test.

About this task

When you initiate a test run that requires a capability such as the Safari browser, and if you have not specified any agent to run, the test enters a **Waiting for agent** state. This state indicates that the job is valid but is waiting for a suitable agent to run it. If an agent that has the required capability is registered with the project, then the agent picks up the job and runs the test, which is in an **Waiting for agent** state. You can also cancel or stop the test runs that are in the **Waiting for agent** state.

Note: The default time limit for the agent to be in the **Waiting for agent** state is 15 minutes after which the test run attempt stops. If you started the test run from the command line, then the time limit is 60 mins.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run, as indicated in the following table:

If...	Then...
You want to run Suites.	Click the **SUITES** tab.
You want to run a Web UI test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.
5. Identify the test asset that you want to run by scrolling through the list, searching for the test, or applying a filter.
6. Click the **Execute** icon in the row of the identified test asset.
   The **Execute test asset** dialog is displayed.
7. Select the version of the test resources that you want to run in the **Version** field.
8. Follow the instructions to select a browser:

   a. Click the **ENVIRONMENT** tab, if it is not already open.

   b. Select an environment from the list that is available for the test run from the **Environment** drop-down list.

The variables and profile overrides in the selected environment override the values defined in the test when you run a test or suite.

c. Select a browser or device profile for the test run from the **Browser** drop-down list.
The browser that is used to record the test is selected as the default browser and the browser specified here overrides the default browser and the browser defined in the profile override.

9. **Optional:** Click the **DATA SOURCES** and select a dataset tab if you are running a test asset that contains datasets.
10. **Optional:** Click the **VARIABLES** tab, and then add or upload variables if the test requires a variable that must be passed to the test at the test run time.
11. Click the **LOCATION** tab, and then select the agent that you have added in the **Override** field.
After selecting the agent, if the specified browser is not available to the agent, Test displays a warning indicator in the **Environment** tab.
12. Click **Execute**.
The test run is initiated.

Results

You have run server test assets on an agent.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Running server test assets by using a cloud credential

You can run tests that are listed as server test assets created in Test on a device cloud by using a cloud credential.

Before you begin

You must have completed the following tasks:

- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.

- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created a Web UI test in Test. See Recording a Web UI test on page 771.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.

About this task

You can run tests on the supported device clouds such as BrowserStack Cloud, LambdaTest Cloud, and pCloudy Cloud.

Restriction: Tests that you record on a standard browser might fail when you play them back on a device cloud. To increase the probability of successful playback, record tests by using device emulation. See supported device profiles on page 771.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select one of the following tabs based on the type of test assets that you want to run:
   - TESTS
   - ADVANCED

   The test assets in the selected tab are displayed.

5. Identify the Server Test asset that you want to run by scrolling through the list, searching for the test, or applying a filter.
   Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.
6. Click the **Execute** icon in the row of the identified test asset.
   The **Execute test asset** dialog is displayed.
7. Select the version of the test resources that you want to run in the **Version** field.
8. Click **Schedule** and select the time if you want to schedule the test run. Else, skip the step.
9. Enter a label for the test run that helps you to identify the test on the **Results** page.
10. Follow the instructions to select a browser:

    a. Click the **ENVIRONMENT** tab, if it is not already open.

    b. Select an environment from the list that is available for the test run from the **Environment** drop-down list.
    The variables and profile overrides in the selected environment override the values defined in the test when you run a test or suite.

    c. Select a device from the list of devices for the configured device cloud in the **Browser** drop-down list.
    The name of the browser is appended with the name of the device and name of the cloud credential set by the user in the *Browser (Full device name : DeviceCloudName set by the user)* format. For example, if the mobile device is *Galaxy S10* and the cloud credential name is *Lambda*, then the name displayed in the list is *Chrome(Galaxy S10 : Lambda)*.

11. **Optional:** Click the **DATA SOURCES** and select a dataset tab if you are running a test asset that contains datasets.
12. **Optional:** Click the **VARIABLES** tab, and then add or upload variables if the test requires a variable that must be passed to the test at the test run time.
13. Click the **RESULTS** tab and select an option if you want to export the test results to a Jira issue in your Xray project in Jira.
    You must have set up Jira as a change management system and configured the Xray integration to export the results.
14. Click **Advanced** to enter any of the following values that must be passed to the test run if they are applicable for the test:
    - **JVM Arguments**
    - **Program Arguments**
    - **Environment Variables**
15. Click **Execute**.
    The test run is initiated.

Results

You have run a server test asset that is created in Test on a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Running server suite assets

You can run a test suite that is listed as server suite asset created in Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Prerequisites to running tests on page 1057, if they apply to the test that you want to configure for a run.
- Read and completed the tasks mentioned in Test run considerations for browser-based Web UI tests on page 1061, if you want to run tests that use Chrome or Edge browsers.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created Web UI tests, and then created a test suite in Test. See Recording a Web UI test on page 771 and Creating a suite on page 830.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.

2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select one of the following tabs to select the test suite that you want to run:
   - SUITES
   - ADVANCED

   The test assets in the selected tab are displayed.
5. Identify the server suite asset that you want to run by scrolling through the list, searching for the test, or applying a filter.
   Clicking the test name displays the **Details** panel. You can view the details of the test asset such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.
6. Click the **Execute** icon in the row of the identified test asset.
   The **Execute test asset** dialog is displayed.
7. Select the version of the test resources that you want to run in the **Version** field.
8. Click **Schedule** and select the time if you want to schedule the test run. Else, skip the step.
9. Enter a label for the test run that helps you to identify the test on the **Results** page.
10. Follow the instructions to select a browser:

    a. Click the **ENVIRONMENT** tab, if it is not already open.

    b. Select an environment from the list that is available for the test run from the **Environment** drop-down list.
    The variables and profile overrides in the selected environment override the values defined in the test when you run a test or suite.

    c. Select a browser or device profile for the test run from the **Browser** drop-down list.
    If you do not specify the browser, the browsers that are configured for individual tests in the test suite are used for the test run.

11. **Optional:** Click the **DATA SOURCES** and select a dataset tab if you are running a test asset that contains datasets.
12. **Optional:** Click the **VARIABLES** tab, and then add or upload variables if the test requires a variable that must be passed to the test at the test run time.

13. **Optional:** Click the **LOCATION** tab, and then select the agent that you have added in the **Override** field.
    After selecting the agent, if the specified browser is not available to the agent, Test displays a warning indicator in the **Environment** tab.
14. Click the **RESULTS** tab and select an option if you want to export the test results to a Jira issue in your Xray project in Jira.
    You must have set up Jira as a change management system and configured the Xray integration to export the results.
15. Click **Advanced** to enter any of the following values that must be passed to the test run if they are applicable for the test:
    - **JVM Arguments**
    - **Program Arguments**
    - **Environment Variables**
16. Click **Execute**.
    The test run is initiated.

Results

You have run a server suite asset that is created in Test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Running server suite assets by using a cloud credential

You can run a test suite that is listed as server suite asset created in Test on a device cloud by using a cloud credential. The tests in the suite run sequentially on device clouds.

Before you begin

You must have completed the following tasks:

- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created Web UI tests, and then created a test suite in Test. See Recording a Web UI test on page 771 and Creating a suite on page 830.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.

About this task

You can run tests on the supported device clouds such as BrowserStack Cloud, LambdaTest Cloud, and pCloudy Cloud.

Restriction: Tests that you record on a standard browser might fail when you play them back on a device cloud. To increase the probability of successful playback, record tests by using device emulation. See supported device profiles on page 771.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select one of the following tabs to select the test suite that you want to run:
    - SUITES
    - ADVANCED

   The test assets in the selected tab are displayed.
5. Identify the server suite asset that you want to run by scrolling through the list, searching for the test, or applying a filter.

   Clicking the test name displays the **Details** panel. You can view the details of the test asset such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.
6. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.
7. Select the version of the test resources that you want to run in the **Version** field.
8. Click **Schedule** and select the time if you want to schedule the test run. Else, skip the step.
9. Enter a label for the test run that helps you to identify the test on the **Results** page.
10. Follow the instructions to select a browser:

    a. Click the **ENVIRONMENT** tab, if it is not already open.

    b. Select an environment from the list that is available for the test run from the **Environment** drop-down list. The variables and profile overrides in the selected environment override the values defined in the test when you run a test or suite.

    c. Select a device from the list of devices for the configured device cloud in the **Browser** drop-down list. The name of the browser is appended with the name of the device and name of the cloud credential set by the user in the *Browser (Full device name : DeviceCloudName set by the user)* format. For example, if the mobile device is *Galaxy S10* and the cloud credential name is *Lambda*, then the name displayed in the list is *Chrome(Galaxy S10 : Lambda)*.

11. **Optional:** Click the **DATA SOURCES** and select a dataset tab if you are running a test asset that contains datasets.
12. Click the **RESULTS** tab and select an option if you want to export the test results to a Jira issue in your Xray project in Jira.

    You must have set up Jira as a change management system and configured the Xray integration to export the results.
13. Click **Advanced** to enter any of the following values that must be passed to the test run if they are applicable for the test:
    - **JVM Arguments**
    - **Program Arguments**
    - **Environment Variables**
14. Click **Execute**.

    The test run is initiated.

Results

You have run a server suite asset that is created in Test on a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of an SAP GUI test

When you want to run an SAP GUI test, then you must configure a run of SAP GUI test or the Compound Test that contains SAP GUI tests on Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Created a Compound Test that contains SAP GUI tests in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test. Refer to Task flow for running SAP tests on DevOps Test Hub.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run the SAP GUI test or Compound test that contains the SAP GUI test in Test, only by using the remote performance agent installed with the SAP runtime environment. If you have not installed any remote performance agent with the SAP run

time environment, you do not see any agents in the **Override** column and the **Execute** button is disabled so that you cannot run the test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
   The test assets in the selected branch of the repository are displayed on the **Execution** page.
3. Perform any of the following steps based on the test you want to run:
   - Click the **TESTS** tab to run the SAP GUI test.
   - Click the **SUITES** tab to run the compound test that contains the SAP GUI test.
4. Identify the test that you want to run and click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

   Note: If you do not want to continue with the current settings, click **Reset** to reset the settings.

5. Select the version of the test resources that you want to run.
6. **Optional:** Click **Schedule** to select a time for scheduling the test run.
7. **Optional:** Click the **Add a label** icon , type a label to identify the test on the **Results** page, and then press the Enter key.
8. Perform the following steps to select the performance agent and run the SAP GUI tests:

   a. Click the **LOCATION** tab, if it is not already open.

      The performance agents installed with SAP GUI runtime are listed under the **Host** column.

      Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

   b. Select the performance agent where you want to run the test asset in the **Override** column.

9. **Optional:** Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

      If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. **Optional:** Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

11. **Optional:** Follow the instructions in this step if the test requires a variable that must be passed to the test at the test run time.

    You must configure the supported browser by using a variable if the test has a browser configured, which is different from the one that is supported by Test.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
    i. Enter `RTW_WebUI_Browser_Selection` as the name.
- To add new variables from your local computer or from the Git repository that is associated with your project, click the **Upload** icon and choose either **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

The default value for the variables is either null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. **Optional:** Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:
    a. Click the **RESULTS** tab.
    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

13. Click **Execute**.

Results

You have started or scheduled a test run of an SAP GUI test or Compound test that contains SAP GUI test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Web UI test to run by using variables

After you added the test resources that you created in the desktop client to the project, you can configure a Web UI test to run by using variables from Test . You must specify the variables for the Web UI test to run on mobile devices that are connected to an agent or a device cloud.

Before you begin

You must have completed the following tasks:

- Created a Web UI test to be run on a mobile device in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Created a variable file if you want to import the variables file. The file must contain the details of the UI Test agent or the device cloud to which the mobile device is connected.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run the Web UI test on a mobile device, emulator, or simulator that is connected to an agent or a device in a device cloud. You must create the test in Test UI, and then add the test asset to your project on Test. You can then run the Web UI test on the devices that are connected to any of the following agents or clouds:

- Remote UI Test agents. Requires the variables at test run time as specified in Variables for the UI Test agent on page 1341.
- The BitBar Cloud. Requires the variables at test run time as specified in Variables for the BitBar Cloud on page 1341.
- The BrowserStack Cloud. Requires the variables at test run time as specified in Variables for the BrowserStack Cloud on page 1341.
- The pCloudy Cloud. Requires the variables at test run time as specified in Variables for the pCloudy Cloud on page 1342.
- The Perfecto Mobile Cloud. Requires the variables at test run time as specified in Variables for the Perfecto Mobile Cloud on page 1342.

You must provide the details of the UI Test agent or the mobile cloud to which the mobile devices are connected as variables in Step 12 on page 1350. You can either enter the variables or use the file in which you entered the variables.

You must refer to the following tables for the variables that are required for a successful run:

Table 611. Variables for the UI Test agent

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use on the UI Test agent. For example, *Chrome(Emulator:Pixel_3a_API_33)*.
appium.server.host	Specify the hostname or IP address of the UI Test agent.
appium.server.port	Specify the port on the UI Test agent that is configured to communicate with Test.

Table 612. Variables for the BitBar Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the BitBar Cloud. For example, *Chrome(BitBar Cloud:Google Pixel 7 Pro)*.
bitbar.apikey	Specify the user token generated for your BitBar Cloud account to authenticate your connection with the BitBar Cloud.
bitbar.host	Specify the hostname of the BitBar Cloud instance.
bitbar.project	Specify the name of the project that contains the recorded test.
bitbar.testrun	Specify a name for the test run that must be displayed in the BitBar Cloud dashboard for the test run.

Table 613. Variables for the BrowserStack Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the BrowserStack Cloud. For example, *Chrome(BrowserStack Cloud:Google Pixel 7 Pro)*.

Table 613. Variables for the BrowserStack Cloud (continued)

Name of the Variable	Action for the Value field
browserstack.host	Specify the URL of the BrowserStack Cloud instance.
browserstack.username	Specify the user name of your BrowserStack Cloud account.
browserstack.apikey	Specify the API key of your BrowserStack Cloud account to authenticate your connection with the BrowserStack Cloud.
browserstack.project	Specify the name of the project that contains the recorded test. **Note:** This variable is optional.

Table 614. Variables for the pCloudy Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the pCloudy Cloud. For example, *Chrome(pCloudy Cloud:Google Pixel 7 Pro)*.
pcloudy.apikey	Specify the API key that is generated for your pCloudy Cloud account to authenticate your connection with the pCloudy Cloud.
pcloudy.username	Specify the username with which you can access the pCloudy Cloud instance.
pcloudy.host	Specify the hostname of the pCloudy Cloud instance.

Table 615. Variables for the Perfecto Mobile Cloud

Name of the Variable	Action for the Value field
RTW_WebUI_Browser_Selection	Specify the web browser and the device to use in the Perfecto Mobile Cloud. For example, *Chrome(Perfecto Mobile Cloud:Google Pixel 7 Pro)*.

Table 615. Variables for the Perfecto Mobile Cloud (continued)

Name of the Variable	Action for the Value field
perfecto.securitytoken	Specify the user token generated for your Perfecto Mobile Cloud account to authenticate your connection with the Perfecto Mobile Cloud.
perfecto.host	Specify the hostname of the Perfecto Mobile Cloud instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **TESTS** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
JMeter	JMeter Test	✓		✓
Ju	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
Se	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.

- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

  Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

  a. Click the **Filter** icon in the **ADVANCED** tab.

     The filter queries that you created and saved are displayed.
  b. Click the filter that you want to apply.

     The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

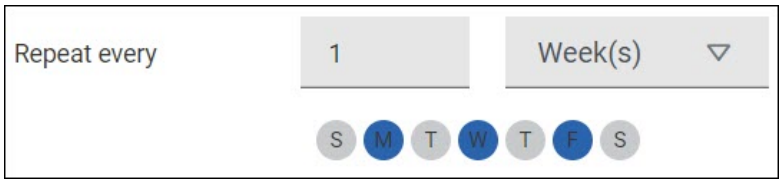
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Perform the following steps to provide the variables that specify the UI Test agent or cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Perform any of the following actions that depend on whether you are using a remote performance agent or a mobile device cloud:
    - Go to Step 15 on page 1352, if you are using an agent.
    - Go to Step 16 on page 1353, if you are using a mobile device cloud.
15. Follow the instructions if you want to run the tests on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:
    a. Click the **LOCATION** tab, if it is not already open.

       The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

       Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

       The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent.

If...	Then...	Action
performance agents are added to the project.		You must wait until the performance agent is available to run the test on the performance agent.
The performance agent that is specified in the test assets is available and there are other performance agents added to the project.	Performance agents that have capabilities that match with the performance agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

16. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Web UI test to run on a device that is connected to an agent or a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.

- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Web UI test to run by using a cloud credential

After you added the test resources that you created in the desktop client to the project, you can configure a Web UI test to run on a device cloud by using a cloud credential from Test .

Before you begin

You must have completed the following tasks:

- Created a Web UI test in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read Test run considerations for using mobile device clouds on page 1069, if you want to configure a run on a mobile device cloud.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.
- Created a cloud credential in your team space that is configured to connect to the mobile device cloud. See Adding cloud credentials of mobile device clouds on page 1119.
- Ensured that you are a member of the cloud credential that is configured to connect to the mobile device cloud. See Viewing details of cloud credentials on page 1126.

About this task

You can run a Web UI test on a device that is connected to a device cloud. You must create the Web UI test in Test UI, and then add the test asset to your project on Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Click the **TESTS** or the **ADVANCED** tab.
5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

   Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

   Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

You can select a parameter from the following list:

- Type
- Test Asset Name
- Test Asset Path
- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

◦ Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

◦ If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

◦ If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, AFT XML files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

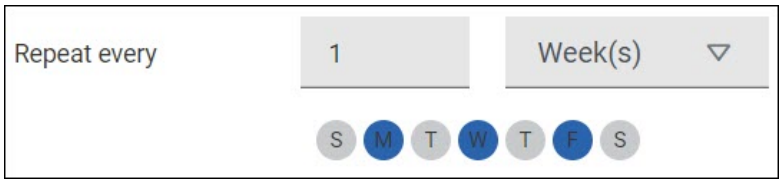
The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.
   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options:

Schedule	Actions
	▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps in the **ENVIRONMENT** tab:

    a. Select the cloud credential of the mobile device cloud from the **Device cloud credentials** list.

       The list displays the cloud credentials that are configured in the team space. The name of the cloud credential is appended to the name of the mobile device cloud in the list. For example, if the mobile device cloud provider is *BitBar* and the name of the credential is *BB_Account1*, then the name displayed in the list is *Bit Bar - BB_Account1*.

       Note: Access to credentials for mobile device cloud services depends on the status of the provider. If you disable a provider on the **Cloud Credentials** page, you cannot access that device cloud provider from the **Device cloud credentials** list.

       If the connection to the mobile device cloud is successful, the **Mobile devices** field is displayed.

    b. Select the devices on which you want to run the tests from the list of devices that are displayed in the **Mobile devices** list.

Note: If you are connected to either the BitBar Cloud or BrowserStack Cloud, the **Projects** field is displayed.

c. Select the project on the mobile device cloud that you want to use or create a project in the **Projects** field. The project overrides the project that is configured in the test asset.

11. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions.

Option	Description
	Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Click **Execute**.
    The test run is initiated.

Results

You have started or scheduled a test run of a Web UI test to run on a device that is connected to a device cloud.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Test run configurations on page 1134

Management of cloud credentials on page 1115

Resetting the configuration settings for a test run on page 1448

Configuring a run of a Windows test

After you added the test resources that you created in the desktop client to the project, you can configure a Windows test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Created a Windows test in HCL DevOps Test UI (Test UI) and added the test asset to the project repository on Test.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

You can run the Windows tests after you create the tests in Test UI, and then add them to your project on Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:
    - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path

- Last Result
- Next Run
- Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, AFT XML files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

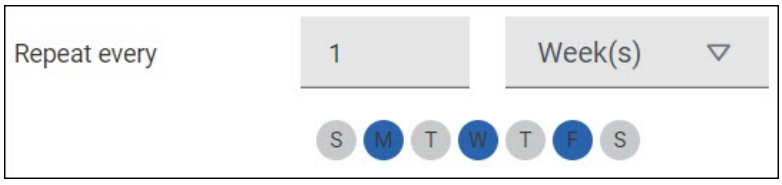
The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.
   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options:

Schedule	Actions
	▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

       If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

    Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

    - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

      Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

    - Select the dataset from the list.

      Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

    - Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

      Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added

datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

12. Perform the following steps to provide the variables that must be passed to the test at the test run time. You can either enter the variables or import the file that contains the variables.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.

        i. Enter `Windows_Device_Selection` as the name.
        ii. Enter `<IP_address>` as the value, which is the IP address of the computer that contains the UI Test agent.

- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

  Note: You must have created a file with the variables before you can select the file.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:

- If you want to run the tests on the remote agents, go to Step 15 on page 1375.
- If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1376.

15. Follow the instructions if you want to run a test on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the performance agent is displayed.

    Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

    The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

    b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	**Then...**	**Action**
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the agent.
The agent that is specified in the test assets is available and other agents added to the project.	Performance agents that have capabilities that match with the agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.

If...	Then...	Action
	the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the performance agent in the **Override** column.	▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

The test run is initiated.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Windows test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a JMeter test

After you added the test resources that you created in JMeter to the server project, you can configure a JMeter test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for JMeter tests on page 1061.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that the JMeter extension on Test is enabled before you configure a test run.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

Important: You can run the JMeter tests only on the default cluster of Test or a remote Kubernetes cluster. You cannot run the JMeter tests on remote agents that are registered with Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:
   - Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the time for scheduling the test run from the following options:

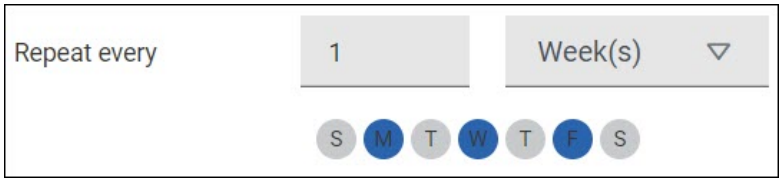
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option.

Schedule	Actions
	You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

   For example, you can set a maximum Java heap size.

b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

For example, enter **-J** to override any JMeter property that is defined in the properties file.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

12. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

    Notes:
    - If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
    - If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

    - Select the **Default Cluster** when no remote clusters are available in your project.
    - Select the remote cluster from the list when a remote cluster is available in your project.
    - Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

    c. Click **Execute**.

    The test run is initiated.

Results

You have started or scheduled a test run of a JMeter test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a JUnit test

After you added the test resources that contain the Maven project with the JUnit tests to the server project, you can configure a JUnit test run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for JUnit tests on page 1063 before you configure a JUnit test run.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that the JUnit extension on Test is enabled before you configure a test run. See Enabling server extensions on page 740.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

Important: You can run the JUnit tests only on the default cluster of Test or a remote Kubernetes cluster. You cannot run the JUnit tests on remote agents that are registered with Test.

After the test runs, you can view the results of the JUnit test in the following formats:

- Surefire
- JUnit XML Report

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the time for scheduling the test run from the following options:

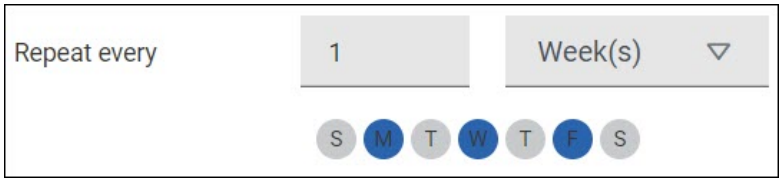
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option.

Schedule	Actions
	You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       You can enter a variable that points to the *MAVEN_OPTS* property.

b. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

   For example, if you have created the JUnit tests by using JDK V12, then you must enter the variable as **JDKOTHER** and the value as **jdkv12**, so that the test run uses the JDK in the path `/data/junit-ext/jdks/jdkv12`.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

12. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

    Notes:
    - If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
    - If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

    - Select the **Default Cluster** when no remote clusters are available in your project.
    - Select the remote cluster from the list when a remote cluster is available in your project.
    - Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

    c. Click **Execute**.

    The test run is initiated.

Results

You have started or scheduled a test run of a JUnit test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Postman test

After you added the test resources that you created and exported from Postman to the server project, you can configure the Postman tests to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for Postman tests on page 1066.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that the Postman extension in Test is enabled before you configure a test run.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

Important: You can run the Postman tests only on the default cluster of Test or a remote Kubernetes cluster. You cannot run the Postman tests on remote agents that are registered with Test.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

8. Select the time for scheduling the test run from the following options:

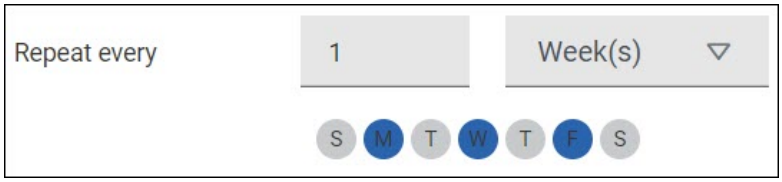
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps:

Schedule	Actions
	a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced**, and then enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

    You can enter the command options that you use when you run Postman tests by using Newman, as the program arguments.

    For example, enter **--verbose**, if you want the details of the test run and each request sent.

    Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

    The default value for each of the fields for the advanced settings is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.

Export options	Results in...	Actions
	The issue is created in your project in Jira after the test run is completed.	
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

12. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

    Notes:
    - If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
    - If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

    - Select the **Default Cluster** when no remote clusters are available in your project.
    - Select the remote cluster from the list when a remote cluster is available in your project.
    - Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

    c. Click **Execute**.

    The test run is initiated.

Results

You have started or scheduled a test run of a Postman test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Rate Schedule or VU Schedule

After you added the test resources that you created in the desktop client to the project, you can configure a Rate Schedule or VU Schedule to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Created Schedules in HCL DevOps Test Performance (Test Performance) and added the test asset to the project repository on Test.
- Read Test run considerations for schedules on page 1067, if you want to configure a run for Rate Schedules or VU Schedules.
- Read Test run considerations for using Machine learning capabilities on page 1064, if you want to configure a run for either a Performance Test or a Schedule that must be analyzed by using Machine Learning capabilities in Test.
- Read and completed the tasks mentioned in Test run considerations for running tests on remote performance agents on page 1073, if you want to use a remote agent as the location to run the test.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Identify the test asset that you want to run by performing any of the following actions:
   - Scroll through the list.

     Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

     Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:

a. Click **New filter**.
b. Enter a name for the filter.
c. Select an operator, and then add a rule or a group of rules.
d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

   You can select a parameter from the following list:
   - Type
   - Test Asset Name
   - Test Asset Path
   - Last Result
   - Next Run
   - Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

   The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

  Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

  a. Click the **Filter** icon in the **ADVANCED** tab.

     The filter queries that you created and saved are displayed.

  b. Click the filter that you want to apply.

     The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

5. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

   Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

6. Select the version of the test resources that you want to run by completing any of the following actions:

Note: The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects that are created in any of the desktop products.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:
- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

7. Select the time for scheduling the test run from the following options:

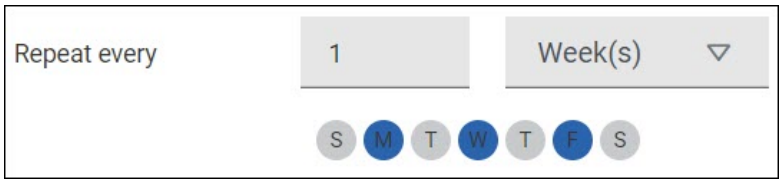
◦ No action is required if you want to initiate the test run immediately after you click **Execute**.

Important: Click **Execute** only after you have configured the other settings in this dialog.

Note: The default time for scheduling a run is **Now**.

◦ Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:

Schedule	Actions
	 e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

8. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

      If you want the test log for the test, you must enter `-history testlog` in the **Program Arguments** field.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

   For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

       The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

       You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list.

       Remember: You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

       - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

         Note: If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

       - Select the dataset from the list.

Note: If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

Important: If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

- Select a data definition from the **Override** list.
- Select one of the following options from the **Generation Type** list:

Option	Description
Standard	Select this option to generate standard test data. Specify the **Number of rows** to generate test data, and optionally, specify the **Seed value**.
Negative	Select this option to generate negative test data to test how the application handles error conditions. Select the **One per row** check box to generate one negative data per row, and optionally specify the **Seed value**.

- Click **Set** to apply the selected data generation settings.
- Click **Execute** if you want to run the test immediately or at the scheduled time. Otherwise, continue with the next step.

11. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

The default value for the variables is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to override the Resource Monitoring labels in the test asset with the Resource Monitoring labels that you created in Test:

    a. Click the **Resource Monitoring** tab.

    Note:

    The **Resource Monitoring** tab displays only if you enabled the **Resource Monitoring from Service** option in Test Performance when you created the test asset.

    b. Click , and press the `Ctrl` + `Space bar` keys, or enter the initial letter of a label to select a label in the list.

    Note:

    You can select or add labels only if you added the labels in the **Resource Monitoring Sources** page in Test.

13. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed.

Export options	Results in...	Actions
	selected issue that exists in your Xray project in Jira.	ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

14. Select any of the following steps that depends on the location to run the test:
    - If you want to run the tests on the remote agents, go to Step 15 on page 1414.
    - If you want to run the tests on a Kubernetes cluster, go to Step 16 on page 1415.
15. Follow the instructions if you want to run a test on the remote performance agent that is connected either to the default Kubernetes cluster or the remote Kubernetes cluster:

    a. Click the **LOCATION** tab, if it is not already open.

    The performance agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the performance agent is displayed.

    Note: You must have added performance agents to your project from the **Agents and Intercepts** page for the performance agents to be displayed under the **Override** column.

    The default value for the performance agents is null or an empty field if no performance agents were configured in the test asset. If the test asset contains performance agents that are configured, then the default performance agent is the first item to be displayed on the list of performance agents listed in increasing alphabetical order.

b. Select the performance agent where you want to run the test asset in the following scenarios:

If...	Then...	Action
The performance agent that is specified in the test assets is available and no other performance agents are added to the project.	The performance agent that is specified in the test assets is displayed in the **Override** column.	Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The performance agent that is specified in the test assets is not available and no other performance agents are added to the project.	There is no performance agent displayed for override in the **Override** column.	You cannot run the test on the performance agent. You must wait until the performance agent is available to run the test on the agent.
The agent that is specified in the test assets is available and other agents added to the project.	Performance agents that have capabilities that match with the agent capabilities specified in the test assets are listed in the **Override** column as follows: ▪ Performance agents with capabilities that best match the capabilities of the agent in the asset are at the top of the list. ▪ Performance agents with capabilities that do not match with the capabilities of the agent in the asset are listed subsequently. You can find details of both performance agents for the matching capabilities when you hover over the performance agent in the **Override** column.	Perform any of the following actions: ▪ Select the performance agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select a performance agent in the **Override** column as the location to run the test.

c. Click **Execute**.

The test run is initiated.

16. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the following remote host to your project that are then displayed under the **Override** column:

Kubernetes clusters that are registered with a team space.

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Rate Schedule or VU Schedule.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuring a run of a Selenium test

After you added the test resources that contain the Maven project with the Selenium tests to a server project, you can configure a Selenium test to be run on Test or a remote Kubernetes cluster.

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Test run considerations for Selenium tests on page 1068 before you configure a Selenium test run.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that the JUnit extension on Test is enabled before you configure a test run. See Enabling server extensions on page 740.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

About this task

Important: You can run the Selenium tests only on the default cluster of Test or a remote Kubernetes cluster. You cannot run the Selenium tests on remote agents that are registered with Test.

After the test runs, you can view the results of the Selenium test in the following formats:

- Surefire
- JUnit XML Report

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.

3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Select the tab based on the type of test assets that you want to run as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules.	Click the **SUITES** tab.
You want to run an individual API test, functional test, or performance test.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, and then run Suites and tests.	Click the **ADVANCED** tab.

   The test assets in the selected tab are displayed.

5. Identify the test asset that you want to run by performing any of the following actions:

   - Scroll through the list.

     Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

     Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests that are in the selected branch of the repository.

- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

  Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

  a. Click the **Filter** icon in the **ADVANCED** tab.

     The filter queries that you created and saved are displayed.
  b. Click the filter that you want to apply.

     The test assets that match the filter criteria are displayed.

You have identified the test asset that you want to run.

Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.

6. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Select the version of the test resources that you want to run by completing any of the following actions:
   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled by the user for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
   - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

     The version that matches the search criteria is displayed and it is selected for the test run.
8. Select the time for scheduling the test run from the following options:
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

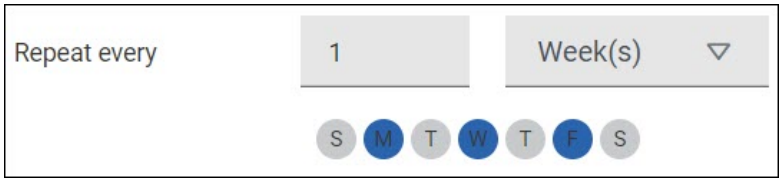
     Note: The default time for scheduling a run is **Now**.
   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date.

<table>
<tr><th>Schedule</th><th>Actions</th></tr>
<tr><td></td><td>c. Select the time at which the test run must start.
d. Select the Never option in the Repeat every field.
e. Click Save.

The test run is scheduled to run at the selected time on the scheduled date.</td></tr>
<tr><td>Recurring test runs</td><td>Perform the following steps:
a. Click the Calendar icon in the row of the Start field.
b. Select the date.
c. Select the time at which the test run must start.
d. Set the frequency at which the test runs must run by entering the number in the Repeat every field, and then select the period from the list. You can select from the following options:
▪ Minute
▪ Hour
▪ Day
▪ Week
▪ Month

For example, if you want the test run to be run every day at the set time, enter <code>1</code> and, then select the Day(s) option.

You can also schedule the test to run at the set time on specific weekdays by selecting the Week(s) option. For example, the following image displays the days selected for the test runs as Monday, Wednesday, and Friday:
 

e. Set the option to end the recurring test runs from the following options:</td></tr>
</table>

Schedule	Actions
	▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

9. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

       You can enter a variable that points to the *MAVEN_OPTS* property.

    b. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

       For example, if you have created the Selenium tests by using JDK V12, then you must enter the variable as **JDKOTHER** and the value as **jdkv12**, so that the test run uses the JDK in the path `/data/junit-ext/jdks/jdkv12`.

    Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

    The default value for each of the fields for the advanced settings is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

12. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have started or scheduled a test run of a Selenium test.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Test run configurations on page 1134

Configuration of test runs in a GitHub Actions Workflow

You can configure test runs of test resources that are in a project repository of Test in a GitHub Actions Workflow. You can choose to run test resources that are in your server project in the GitHub Workflow when you do not want to run the test resources without opening the Test project.

Before you configure a test run for any test resources that are in a project repository of Test, you must complete certain tasks. See Prerequisites to run tests in a GitHub Actions Workflow on page 1426.

After you complete the prerequisite tasks, you can proceed to configure a test run in a GitHub Workflow. See Configuring a test run in a GitHub Actions Workflow on page 1427.

Prerequisites to run tests in a GitHub Actions Workflow

Before you configure a test run for any test resources that are in a project repository of Test , you must complete certain tasks to run the test resources in a GitHub Actions Workflow.

You must be familiar with creating projects and running test resources from Test.

You must complete the following tasks to set up and use Test:

- Install Test. See Installation of DevOps Test.
- Possess valid licenses to use Test. See Management of licenses in DevOps Test.
- Create a project on Test. See Creating a project on page 1728.
- Add a repository that contains test resources to the project repository. See Adding repositories to a server project on page 1729.
- Generate the Offline User Token on Test that is required for connecting to Test from another application such as the GitHub Action. See Access management for DevOps Test on page 1709.
- Open the project to view the details of the test resources that you want to run in a GitHub Actions Workflow. You must make a note of the test resource details from the **Execution** page of Test.

  You must note the following details of the test resource that are to be provided in the YAML file for the GitHub Action:
  - The URL of the Test instance.
  - The name of the team space that contains the project on Test.
  - The name of the project on Test that contains the test resources.
  - The branch name of the project repository that contains the test resources.
  - The ID of the test resource.

You must complete the following tasks outside of Test:

- Create an empty Git repository on any GitHub Enterprise server, if you do not already use Git repositories to store the test resources that you create. For more information about creating Git repositories, refer to Creating a new repository.

  If you already use Git repositories to store test resources created, you can use the same Git repositories.

Note: You must have the required privileges in the Git repository to create a GitHub Actions Workflow.

- Enable the GitHub Actions option in the Git repository, if it is not available as an option to use. For more information about enabling the GitHub Actions, refer to Learn about GitHub Actions.

You can continue to configure a run of a test, see Configuring a test run in a GitHub Actions Workflow on page 1427.

Configuring a test run in a GitHub Actions Workflow

After you complete the prerequisite tasks to run tests in a GitHub Actions Workflow, you can configure the test run in a GitHub Actions Workflow. The test is in a project on Test .

Before you begin

You must have completed the following tasks:

- Completed the tasks that are mentioned in Prerequisites to run tests in a GitHub Actions Workflow on page 1426.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Extended the CA list. See Extending the trusted certificate authority list on Linux on page 735 or Extending the trusted certificate authority list on Windows on page 736.

1. Perform the following steps to open the Git repository that you use or created, which has the GitHub Actions option enabled:
    a. Open the Git repository on your GitHub Enterprise server.
    b. Create a folder with the name as `.github/workflows` in the root folder of the repository.
    c. Create an empty YAML file in the `.github/workflows` folder.

    The YAML file is used to define the GitHub Action in the Actions Workflow.
2. Perform the following steps to define the YAML file for the GitHub Action:
    a. Open the GitHub Marketplace (https://github.com/marketplace?type=) in a web browser.
    b. Search for *HCL DevOps Test Hub GitHub Action*.
    c. Open the GitHub Action.
    d. Read through the information provided.
    e. Copy the contents of the YAML file that is provided in the example by clicking the copy icon .
    f. Open the YAML file that is in the `.github/workflows` folder of your repository and paste the copied contents.

g. Enter the valid details as values for the required attributes in the YAML file by following the actions in the following table:

Attribute	Action
Required attributes	
serverURL	Specify the URL of the Test instance.
offlineToken	Specify the Offline User Token that is generated by Test.
teamspace	Specify the name of the team space that contains your project on Test.
project	Specify the name of your project on Test.
branch	Specify the name of the branch in the repository that contains the test resources, which can be found in the **Details** pane of the test resource on the **Execution** page.
assetId	Specify the ID of the test resource, which can be found under the **Git details** section in the **Details** pane of the test resource on the **Execution** page.
Optional attributes	
environment	Specify the environment that is configured in the test.
datasets	Specify the dataset that must be used at test run time, which replaces the dataset that is configured in a test.
labels	Specify text labels to identify a specific test result when the test run is complete.
secretsCollection	Specify the name of the secret collection that is configured in a test.
variables	Specify the name of the variable and its value if your test requires variables during the test run time. **Note:** You can add multiple variables that are separated by a semicolon.

h. Save the YAML file.

3. Perform the following steps to set up a Runner for the GitHub Action in GitHub:

a. Click the Settings icon .

b. Click **Actions > Runners > New self-hosted runner**.

c. Follow the on-screen instructions to set up a self-hosted runner on your computer.

After the runner is installed on your computer, you must verify that the runner is ready to run jobs in the GitHub Actions Workflow and is in the *Active* state. The runner is listed in the **Runners** pane in the Git repository.

4. Perform the following steps to start the GitHub Action in the Workflow:

a. Click the **Actions** option in the Git repository.

The GitHub Action that you configured in the YAML file for Test is displayed as an **Action** under **Workflows**.

b. Click the *HCL DevOps Test Hub GitHub Action*, and then click **Run workflow**.

Results

You have configured and run a resource that is in your project on Test in a GitHub Action Workflow.

What to do next

You can perform any of the following tasks on the Git repository page by clicking the job that was completed:

- View the status of the test run by expanding the **Execute test** section.
- View the links to the test results and reports listed in the **Execute test** section.

You can perform any of the following tasks from Test:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Configuration of a scan of an application code

You can use Test to scan your application code for security vulnerabilities similar to using the HCL® AppScan CodeSweep plugin to scan code in Visual Studio.

Test is enabled to support scanning of your application code that is in a remote repository.

Before you commit your application code to the remote repository, you must complete certain tasks. See Considerations to scan an application code on page 1430.

After you commit the assets to the remote repository, you must complete the following tasks:

1. Log in to Test.
2. Open the project in your team space.

3. Add the remote repository to your project.
4. Open the **Execution page**, and then run a scan of the asset.

For detailed instructions, see Configuring a run to scan the application code on page 1432.

Considerations to scan an application code

When you want to use Test to run a scan of the application code for security vulnerabilities, you must read the considerations that you must take into account and complete certain tasks.

Before you can run a scan of the application code on Test, you must have completed the following tasks:

- Created a file that has the following characteristics:
  - The format of the file must be `yaml`.
  - The name must be `.codesweep`.
- Committed the following assets or resources to the remote repository:
  - The `.codesweep` file.
  - The application code can either be in the root directory or another folder in the root directory.

  Note: The default root directory is the directory that contains the `.codesweep` file.

You can specify the following information in the `.codesweep` file:

- The name of the asset that must be displayed as the test asset in the **Execution** page of Test. The name must be specified at the first level in the file as the value of the **name** attribute.
- Optionally, the name of folders that contain the application code.

  All files and folders in the root directory are scanned as the default action, if no files or folders are specified under the **folders** attribute.

  You can specify the files or folders to scan by using any of the following options:
  - Specify the files or folders by using an `XML` file that is in the same format as the HCL AppScan IRX file.
  - Specify the files or folders in the `.codesweep` file under the **includes** or **excludes** attributes under the **folders** attribute.

- Optionally, you can specify the name of files or folders that must be included in the scan. You must specify the files or folders as the value of the **includes** attribute under the **folders** attribute.
- Optionally, you can specify the name of files or folders that contain the application code that need not be scanned. You must specify the files or folders as the value of the **excludes** attribute under the **folders** attribute.

Note: You cannot use *wildcard* characters to indicate the files or folders anywhere in the `.codesweep` file.

- Optionally, you can specify the rules that are to be ignored in the scan. You can specify the rules by using any of the following options:
  - Specify a file, which is a copy of `~/appscan/custom_scanners_service_ignore.json` that is created by the CodeSweep plugin for Visual Studio. With this method, you can reuse the preferences that you have configured in your development environment.
  - Specify a list of rules. Each rule is defined by a name and a language identifier.
- Optionally, you can specify the criteria for attributes and their values that specify the verdict of the scan and the range for the security errors in the code.

For example, you can consider the following details:

Where...	Is...
The name of the asset that is to be displayed on the **Execution** page	*mycode*
The folder that contains the code to be scanned	*use_code*
The folder that contains the code that is to be excluded in the scan	*do_not_use_code*
The value of the attribute *Verdict*	
The range for the security errors is classified as follows: • *max-high* • *max-medium* • *max-low*	The conditions for a *pass* verdict are as follows: • *0* • *10* • *100*

The contents of the `.codesweep` file is as follows:

```
name: mycode
folders: # default=./ , Optional
    configuration: configuration.xml # XML Configuration file, Optional
    includes: # list of folders to include, Optional
        - use_code
    excludes: # list of folders to exclude, Optional
        - do_not_use_code
```

```
ignore: # Ignore rules, Optional
    file: custom_scanners_service_ignore.json # json file coming from visual code, Optional
    rules: #list of rules to ignore, Optional
        - name: com.ouncelabs.languagelite.nodejs.Rules.NodeJS_PrivateDataExposedviaHTTPURL # Name, Required
          language: NodeJS # Language, Required
        - name: com.ouncelabs.languagelite.nodejs.Rules.NodeJS_PrivateDataExposedviaHTTP # Name, Required
          language: NodeJS # Language, Required
verdict: # PASS verdict, Optional
    max-high: 0 # default=0, Optional
    max-medium: 10 # default=0 , Optional
    max-low: 100 # default=-1 (no limit), Optional
```

You can continue to configure a run of a scan of the code, see Configuring a run to scan the application code on page 1432.

Related information

Configuration of a scan of an application code on page 1429

Configuring a run to scan the application code

After you added the assets in the remote repository to a project in your team space, you can configure a run to scan the application code specified in the `codesweep` file by using the execution service on Test .

Before you begin

You must have completed the following tasks:

- Read and completed the tasks mentioned in Considerations to scan an application code on page 1430.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Completed the Xray integration, if you want to export the test results to a Jira issue in your Xray project in Jira. See Integration with Xray test management for Jira on page 1703.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Added the remote Git repository that contains the test resources to the project. See Adding repositories to a server project on page 1729.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the test assets, and then click **Execution**.
3. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

4. Identify the test asset or resource that you want to run by performing any of the following actions:
    - Scroll through the list.

    Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

    Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule, or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

e. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

5. Click the **Execute** icon in the row of the identified test asset.

The **Execute test asset** dialog is displayed.

Notes:

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

6. Select the version of the test resources that you want to run by completing any of the following actions:

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

Use the following details about the version of the test resources that are displayed to identify the version that you want:

- Commit message.
- Tags labeled by the user for the version committed.
- The user who committed the version to the repository.
- Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch of the repository. If you do not select any version, then the latest version is selected for the test run.

Notes:

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon. As a result, the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you chose for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

7. Select the time for scheduling the test run from the following options:

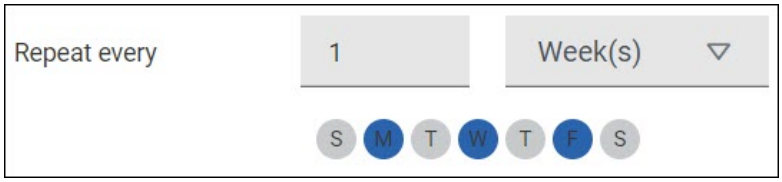
   - No action is required if you want to initiate the test run immediately after you click **Execute**.

     Important: Click **Execute** only after you have configured the other settings in this dialog.

     Note: The default time for scheduling a run is **Now**.

   - Select **Schedule** and perform the following actions if you want to schedule a single test run or configure recurring test runs:

Schedule	Actions
A single test run	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start. d. Select the **Never** option in the **Repeat every** field. e. Click **Save**. The test run is scheduled to run at the selected time on the scheduled date.
Recurring test runs	Perform the following steps: a. Click the **Calendar** icon in the row of the **Start** field. b. Select the date. c. Select the time at which the test run must start.

Schedule	Actions
	d. Set the frequency at which the test runs must run by entering the number in the **Repeat every** field, and then select the period from the list. You can select from the following options: ▪ Minute ▪ Hour ▪ Day ▪ Week ▪ Month For example, if you want the test run to be run every day at the set time, enter `1` and, then select the **Day(s)** option. You can also schedule the test to run at the set time on specific weekdays by selecting the **Week(s)** option. For example, the following image displays the days selected for the test runs as *Monday*, *Wednesday*, and *Friday*:  e. Set the option to end the recurring test runs from the following options: ▪ Select the **Never** option, if you do not want the recurring test runs to end. ▪ Select the **On** option, and then click the **Calendar** icon. You can select the date and time after which the scheduled test runs do not run. f. Click **Save**. The recurring test runs are scheduled to start the first run at the selected time on the scheduled date, and to repeat the run at the frequency that you set.

8. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   Important: The configuration that you set for the test run in the **Execute test asset** dialog is preserved when you run the same test again. Those changes are not visible when another user logs in to Test. For example, if

you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you want to export the test results to a Jira issue in your Xray project in Jira:

    a. Click the **RESULTS** tab.

    b. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: i. Select the **Jira Xray** option. The **Select issue** field is displayed. ii. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. iii. Select the issue to which you want to export the test results. iv. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: i. Select the **New Jira item** option. ii. Continue with the next step.
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ▪ You do not want to export the test results. ▪ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. i. Select the **None** option. ii. Continue with the next step.

10. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the test asset.

c. Click **Execute**.

The test run is initiated.

Results

You have configured and either started or scheduled a run of a scan of the application code that is specified in the `codesweep` file.

What to do next

You can perform any of the following tasks:

- Stop the test run at any point after the test run is initiated from the **Execution** page. See Stopping test runs on page 1449.
- Cancel a scheduled test run from the **Execution** page. See Canceling scheduled test runs on page 1452.
- View all the states of the test asset by clicking the **Show in the Progress page** icon for the test asset for which you started or scheduled the run. See Viewing the state of test assets on page 1440.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 1444.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 1499.

Related information

Resetting the configuration settings for a test run on page 1448

Configuration of a scan of an application code on page 1429

Test run configurations on page 1134

Management of running tests

Find information about the tasks that you can perform on a test that you configured for a run either while it runs or after it completes the run.

Viewing the state of test assets

After you initiate or schedule a test run on the **Execution** page, you can view the state of the test assets on the **Progress** page by clicking the **Show in the Progress page** icon that is enabled for that test asset on the **Execution** page. You can view all the states of the test asset from the past hour such as `In transition`, `Running`, `Scheduled`, `Completed`, or `Stopped`.

Before you begin

You must have initiated runs of the test assets in your project from the **Execution** page.

Note: The **Show in the Progress page** icon is enabled for the test asset only if a run of the test asset is either started or scheduled.

1. Click **Execution**, if you are not already on the **Execution** page.
2. Select the branch of the repository that contains the test assets.

   The test assets that are contained in the selected branch of the repository are displayed in the following tabs on the **Execution** page:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

3. Identify the test asset or resource that you want to run by performing any of the following actions:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule	✓		✓

- Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** field. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all functional tests in the selected branch of the repository.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then add a rule, or a group of rules.
  d. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - Type
     - Test Asset Name
     - Test Asset Path
     - Last Result
     - Next Run
     - Components

  e. Save the filter query to save and apply the filter query to filter the assets based on the query.

     The test assets that match the filter criteria are displayed.
- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

a. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

b. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

4. Click the **Show in the Progress page** icon in the **Actions** column of the test asset.

The **Progress** page is displayed with the following details:

- The different states of the test asset runs that were initiated or completed in the past hour are displayed as follows:
  - `In transition`
  - `Running`
  - `Completed`
  - `Stopped by User`
  - `Canceled`
- The ID of the selected test asset is applied as a filter and is displayed in the filter row.

  The applied filter is removed along with the displayed items for this filter when you perform any of the following actions on the **Progress** page:
  - You click **Hide inactive**, and then click **Show all**.
  - You visit any other page in the Navigation pane and return to the **Progress** page.

Results

You have viewed all the states of a specific test asset runs that were initiated during the past hour.

What to do next

You can choose to perform any of the following tasks from the **Progress** page:

- Stop test assets that are running. See Stopping test runs on page 1449.
- Cancel scheduled runs. See Canceling scheduled test runs on page 1452.
- View the results or the execution log of the test asset from the **Actions** column by clicking the **Open action menu** icon . See Checking logs on page 1447.
- View the results, reports, and logs of the completed test run from the **Results** page. See Test results on page 1499.

Viewing the progress of running test assets

After you initiate or schedule a test run on the **Execution** page, you can view the progress of the test assets that are running or scheduled to run, from the **Progress** page. You can also view the test assets that have completed their run, are stopped or cancelled, during the past hour.

Before you begin

You must have initiated runs of the test assets in your project from the **Execution** page.

1. Open your project and click **Progress**.

   Any one of the following views is displayed:
   - No test assets are displayed in the **Progress** page if there are no test assets that are running, scheduled to run, or have completed their run in the past hour.
   - Test assets in your project that are in any of the following states are listed:
     - Running
     - Scheduled
     - Completed
     - Stopped
     - Canceled
   - No test assets are displayed but the option **Show all** is enabled. You can click **Show all** to display the test assets that ran in the past hour. Test assets that were hidden from the display when **Hide inactive** was clicked are displayed.

   Note: Test assets are automatically removed from the display after *60 minutes* of being added to the **Progress** page. You can hide the test assets from the display by clicking **Hide inactive**, so that you can view only the test assets that were added to the **Progress** page in the past hour. You cannot use the **Progress** page to view the history of test assets that were run from the **Execution** page.

2. Identify the test asset by performing any of the following actions:
   - Identify the test asset or resource by scrolling through the list.
   - Search for the test asset by entering any text contained in the test asset name in the **Search** text box. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
   - Create a filter query by using the **New filter** option by performing the following steps:
     a. Click **New filter**.
     b. Select an operator, and then add a rule, or a group of rules.
     c. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

        You can select a parameter from the following list:

- Type
- Name
- Started by
- Started on
- Finished on
- Status
- Verdict

d. Apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

e. Save the filter query by performing the following steps, if you want to reuse the filter query later:

i. Click **Save**.
ii. Enter a name for the filter query.
iii. Click **Save**.

◦ Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

Note: To open the filter query, you must have created and saved a filter query.

a. Click the **Open filters** icon .
b. Select the saved filter in the **Filters** dialog.
c. Click **Apply** to apply the filter.

The test assets that match the filter criteria are displayed.

◦ Sort the test assets by the user who started the test by clicking the **Started by** column header, and find the test asset listed against your name.
◦ Sort the test assets by the type of the test assets by clicking the **Type** column header, and find the test asset type you ran or scheduled.
◦ Click **Hide inactive** to only display the test assets that are running or scheduled to run. This action removes all completed, canceled, stopped, or failed test assets from the display.

Note: The items in the rows are sorted in an ascending or descending order when the column header is clicked. The following icons show the status of sorting in a column:

- items in the column are not sorted.
- items in the column are sorted in the ascending order or with the earliest record on top.
- items in the column are sorted in the descending order or with the latest record on top.

You can view the following details of the test assets on the **Progress** page:

Column header	Description
Type	Displays the icon for the type of the test asset.
Name	Displays the name of the test asset added to the project from the Git repository. Clicking the test name displays the **Details** panel. You can view the details of the test such as the description, the branch in the repository that contains the asset, the Git details, and the details of the commits to the repository. You can also view the history of the test runs for the specific test under the **History** tab of the **Details** panel.
Started by	Displays the name of the user who started the run of the test asset.
Started on	Displays the date and time when the test run was started.
Finished on	Displays the date and time when the test run completed the run.
Status	Displays the state of the test asset progressively. For example, after the run is in the `Initiated` state, it moves on to the `Running` state and ends in its final state depending on the verdict of the test run as either `Completed`, `Failed`, or `Inconclusive`. If the test run is stopped or a scheduled run is canceled, the status displayed is `Stopped by User` for a stopped run or `Canceled` for a canceled test.

From the **Actions** column, you can perform the following operations on your test asset:

- Stop a run.
- Cancel a scheduled run.
- Cancel scheduled recurring test runs.
- View the test log.
- View the result of the test asset on the **Results** page.
- Query the test result data.

Results

You have viewed the progress of the test assets that you ran from the **Execution** page.

What to do next

You can choose to perform any of the following tasks from the **Progress** page:

- Stop test assets that are running. See Stopping test runs on page 1449.
- Cancel scheduled runs. See Canceling scheduled test runs on page 1452.
- View the results or the execution log of the test asset from the **Actions** column by clicking the **Open action menu** icon . See Checking logs on page 1447.
- View the results, reports, and logs of the completed test run from the **Results** page. See Test results on page 1499.
- Query the test results data. See Test log queries on page 1538.

Checking logs

To verify how the test ran or to debug test run failure, you can check the `Test Log` and `Execution log`.

About this task

The `Test Log` displays the interaction between Test and the application or system under test. After the test run completes, the verdict of the run can be *Pass*, *Fail*, or *Inconclusive*. If the verdict of the run is Pass, the `Test Log` is available in the **Results** page.

The `Execution log` displays the console messages of the run time process that runs the test. This log is useful in determining the cause of the failure if the verdict of the run is *Fail* or *Inconclusive*. You can view the `Execution log` from the **Progress** page.

Viewing the `Test Log`

1. Go to the **Results** page and identify the test that you ran.
2. Click the test so that the **Reports** panel is displayed.
3. Click the **Test Log** in the **Reports** panel.

   The `Test Log` is displayed in a browser window.

Viewing the `Execution` log

4. Go to the **Progress** page and identify the test that is in the *Running* state.
5. Click the **Open action menu** icon.

6. Click **Execution log**.

   The `Execution log` is displayed in a browser window.

Related information

Test run configurations on page 1134

Resetting the configuration settings for a test run

When you are configuring a test run either as a first-run or when repeating the run, and you do not want to proceed with the settings configured or saved for the test run, you can reset the configuration settings. Resetting the configuration reverts all the settings to their default values for the test run.

Before you begin

You must be a member of the project with the *Owner* or *Tester* role to run the tests.

You must ensure that you have the **Execute test asset** dialog is displayed before you proceed with resetting the configuration settings for the test run.

About this task

You can reset the configuration settings when you are configuring a test run either for its first run or when you are repeating the test run.

1. Click **Reset** in the **Execute test asset** dialog at any point when you want to reset the configuration settings for the test run.

   Notes:

   - If you have configured some or all of the settings for the current test run and you do not want to continue with those settings, clicking **Reset** resets the settings to their default values.
   - If you are repeating a test run and do not want to use the saved settings from a previous run, clicking **Reset** reverts all the saved settings to their default values.

   The settings in the **Execute test asset** dialog are reset to their default values.
2. Use the following table to find the default values for each of the settings in the **Execute test asset** dialog:

Window or Tab	Field or Setting	Default value is
Execute test asset	Scheduling the test run	**Now** is selected.
Execute test asset	Label for settings	Null or empty field.
Advanced	**JVM Arguments**	Null or empty field.

Window or Tab	Field or Setting	Default value is
	Program Arguments	Null or empty field.
	Environment Variables	Null or empty field.
ENVIRONMENT tab	API test environment	The environment configured in the test asset.
	Secrets collection	Null or empty field.
DATA SOURCES tab	**Override** list	Null or empty field if the asset has no dataset.
		The dataset in the test asset if the asset has one dataset.
		The first dataset on the list of datasets, which are listed in increasing alphabetical order, if the asset has multiple datasets.
VARIABLES tab		Null or empty field.
LOCATION tab	Agents	Null or empty field if the asset has no agents configured.
		The first agent on the list of agents, which are listed in increasing alphabetical order, if the asset has multiple agents configured.

What to do next

You can complete configuring the settings that you want for the test run. See Test run configurations on page 1134.

Stopping test runs

You might want to stop a test run or multiple tests that are running when you realized that you did not configure all the settings or you want to change a few settings for the test runs.

Before you begin

You must have initiated a run for one or multiple test assets from the **Execution** page.

About this task

You can stop either a running test or multiple running tests from the **Progress** page. You can stop the test if it is in either the `In transition` or `Running` state. You cannot stop the test run if the test has already completed its run.

1. Go to the **Progress** page and identify the tests in any of the following ways that you want to stop:

   You can select the tests that satisfy the following conditions:

- The test must be either in the `In transition` or `Running` state.
- The **Stop execution** icon in the **Actions** column must be enabled for the test.
- The checkbox must be enabled for the test.

2. Perform the actions in the following table:

If...	Action...
You want to stop any test that is running	Perform any of the following actions: ◦ Click the **Stop execution** icon in the **Actions** column of the selected test. ◦ Click the checkbox in the row of the selected test, and then click **Stop selected**. The **Stop execution** dialog is displayed.
You want to stop multiple tests that are running	Perform any of the following actions: ◦ Click the checkbox in the row of the selected tests, and then click **Stop selected**. ◦ Click the **Select all** checkbox in the header row, and then click **Stop selected**. **Note:** All running tests are selected when you click the **Select all** checkbox. The **Stop execution** dialog is displayed.

3. Perform the following steps in the **Stop execution** dialog that is displayed:

   a. Set the timeout period for stopping the test run. Enter a numeric value and select the unit from the options available as *Seconds*, *Minutes*, or *Hours*.

   The time out period is the time during which the test run is allowed to stop on its own and after the timeout period, the test is forced to stop abruptly.

   Important: If the timeout period is not set, a value of *30 seconds* is considered as the default timeout period.

   b. Perform any of the following actions:

   - Keep the **Capture results** option selected, if you want the results to be captured for the test.

     The captured results for the test are available to view from the **Results** page.
   - Clear this option if you do not want to capture the results.

Note: The **Capture results** option is selected as the default option.

Important: The results are always captured for API suites even if you clear the **Capture results** option.

c. Perform any of the following actions:

- Keep the **Execute 'finally-block', if present** option selected, if you want the *finally-block* code in the test script, if present, to be run before the test run is stopped.
- Clear this option if you do not want to run the *finally-block* code in the test script.

Note: The option **Execute 'finally-block', if present** is selected as the default option.

Restriction: The *tear-down* steps equivalent to the **'finally-block'** code, if present in API suites are not controllable at test runtime. The **Execute 'finally-block', if present** option has no impact on API suites.

d. Click **Stop execution**.

A notification is displayed that the running tests are stopped.

Note: If the test run completes before you can configure the options for stopping the test run, a notification is displayed in the **Stop execution** dialog that the test asset has completed its run.

Results

You have successfully stopped a running test or stopped multiple running tests. The stopped test runs are displayed on the **Progress** page with the status as *Stopped by User*. Stopped tests are not considered in the count of tests executed that are displayed on the **Overview** page.

What to do next

You can re-initiate the test run from the **Execution** page or from the **Results** page by completing the configurations that you want for the test.

You can view the results or the execution log of the stopped test from the **Actions** column on the **Progress** page by clicking the **Open action menu** icon .

Related information

Test run configurations on page 1134

Viewing the progress of running test assets on page 1444

Canceling scheduled test runs on page 1452

Canceling scheduled test runs

You might want to cancel a scheduled test run, scheduled recurring test runs, or multiple scheduled test runs when you realized that you did not configure all the settings for the runs, want to change a few settings, or do not want those tests to run.

Before you begin

You must have scheduled runs for one or multiple test assets from the **Execution** page.

About this task

You can cancel a scheduled test run, all instances of a scheduled recurring test runs, or multiple scheduled tests from the **Progress** page.

1. Preform any of the following actions:
   - Click **Scheduled** under the **Next Run** column on the **Execution** page.

     The test is displayed on the **Progress** page. If you selected a test that was scheduled as a recurring test, then all the tests that ran and the next scheduled test run are displayed.

     Go to Step 2 on page 1452.
   - Go to the **Progress** page and identify the tests in any of the following ways that you want to cancel:

     You can select the tests that satisfy the following conditions:
     - The test must be in the `Scheduled` state.
     - The **Cancel** icon ⊗ in the **Actions** column must be enabled for the test.
     - The checkbox must be enabled for the test.

     Go to Step 2 on page 1452.
2. Perform the actions in the following table:

If...	Action...
You want to cancel any test run that is scheduled.	Perform any of the following actions: ◦ Click the **Cancel** icon ⊗ in the **Actions** column of the test that displays *Scheduled* as the status. ◦ Click the checkbox in the row of the test that is scheduled, and then click **Stop selected**. The **Cancel scheduled execution** dialog is displayed. Go to Step 3 on page 1453.
You want to cancel all scheduled instances of the recurring test runs.	
You want to stop multiple test runs that are scheduled	Perform any of the following actions:

If...	Action...
	◦ Click the checkbox in the row of the selected tests, and then click **Stop selected**. ◦ Click the **Select all** checkbox in the header row, and then click **Stop selected**. **Note:** All scheduled tests are selected when you click the **Select all** checkbox. The **Cancel scheduled execution** dialog is displayed. Go to Step 3 on page 1453.

3. Click **Yes** in the **Cancel scheduled execution** dialog.
   Result
   .

   A notification is displayed that the scheduled test runs are canceled.

Results

You have successfully canceled a scheduled test, all instances of the scheduled recurring tests, or multiple scheduled tests. The canceled test runs are displayed on the **Progress** page with the status as *Canceled*. The canceled test runs are not displayed on the **Results** page and are not considered in the count of tests executed that are displayed on the **Overview** page.

What to do next

You can re-initiate the test run from the **Execution** page by completing the configurations that you want for the test.

Related information

Test run configurations on page 1134

Viewing the progress of running test assets on page 1444

Stopping test runs on page 1449

Management of virtualized services

You can find information about the tasks that you can perform on and manage the virtualized services that run on Test . You can start or stop the virtual services that are available to Test. You can view the routing rules and usage statistics of virtualized services, agents, or intercepts that are connected to Test.

You can create stubs or virtual services in HCL DevOps Test Integrations and APIs (Test Integrations and APIs). You must commit the virtual service resources to a remote Git repository, and then add the Git repository to your project on Test if you used Test Integrations and APIs to create virtual service resources.

You can start a running instance of the virtual services from Test to run on the following locations:

- A registered API agent.
- The default Kubernetes cluster of Test.
- A remote Kubernetes cluster.

You can run all virtual services on registered API agents except the virtual services that use CICS transport, DB2/Z transport, or Istio.

The default Kubernetes cluster or a remote Kubernetes cluster supports running of the following types of virtual services:

- Virtual services that utilize the WebSphere® MQ transport.
- Virtual services that utilize the HTTP transport.

If running a virtual service in a Kubernetes cluster, the following are the considerations that decide the selection of a container:

- Before a container is selected to run a virtual service, it is checked for a previous failure to start a virtual service or for a low heap size, and other conditions. If the container was not found suitable, then it is not selected for running the virtual service.
- Multiple virtual services can be run in the same container if the following conditions are satisfied:
  - Subsequent virtual services are started by the same project member.
  - All virtual services are created and contained in the same Test Integrations and APIs project.
  - The requests to run the virtual services refer to the same version, meaning that the git commit IDs on which they are based are the same. If you use the UI, this means that the **Version** field in the **Execute virtual service** dialog are the same.
  - The settings that you configure in the **Advanced** settings for the virtual services are the same.
  - The datasets selected for an override are the same, if datasets are used.

Working with virtual services

You can perform the following tasks on virtual services on Test:

- Read about the considerations before you configure a run of the virtual services. See Prerequisites for running HTTP virtual services on page 1455.
- Read about the considerations before you configure a run of the virtual service on an API agent. See Considerations for running virtual services on remote API agents on page 1081.
- Set up the HTTP proxy, if you have configured stubs in the test assets to use an HTTP proxy to route requests. See Setting up intercepts on page 1457.

- View all intercepts that are registered with Test including those that you have not configured. See Viewing intercepts that are registered with a team space on DevOps Test on page 1459.
- View virtual service resources in the test assets that are in the repositories added to a project on Test. See Viewing virtual service resources on page 1460.
- Configure runs of virtual services. See Configuring a run of a virtual service on page 1462.
- Configure runs of HTTP virtual services to run without using proxies. See Running HTTP virtual services in Kubernetes cluster without using proxies on page 1477.
- View running instances of virtual services. See Viewing running instances of virtual services on page 1482.
- View configurations of a running virtual service instance. See Viewing configurations of running instances of virtual services on page 1487.
- Modify the configurations of a running virtual service instance. See Modifying configurations of running instances of virtual services on page 1492.
- View details of the usage statistics of the virtual services. See Viewing usage statistics of virtual services on page 1496.
- View routing rules of the intercepts. See Viewing routing rules of the virtual services on page 1494.
- Stop a running virtual service instance. See Stopping virtual services on page 1498.

Prerequisites for running HTTP virtual services

You can find information about the prerequisite tasks that you must complete before you configure a run of the HTTP virtual services in Test .

Prerequisites for running HTTP virtual services in a Kubernetes cluster

You can find information about the tasks that you must complete before you configure a run of the HTTP virtual services in the Test cluster or a remote Kubernetes cluster.

Important: The prerequisites do not apply when you use an API agent to run the HTTP virtual services.

Conditions for running of HTTP virtual services

Before you run the HTTP virtual services in the Test cluster or a remote Kubernetes cluster, you must check for the following conditions and perform the actions indicated:

If...	Then...
You want to route the HTTP traffic via the HTTP proxy either to the virtual service or live systems, based on routing rules.	You must set up the HTTP proxy. See Setting up intercepts on page 1457.
You want to route the HTTP traffic to the virtual service via the HTTP proxy and the proxy is not configured to trust all certificates.	The trust store must be configured to include the certificate authority for the server certificate. See Certificate authority: Importing and extending lists on page 732.

If...	Then...
A client application is sending requests directly to the virtual service.	The client application must trust the certificate authority for the server certificate. See Certificate authority: Importing and extending lists on page 732.

You are working with Test V10.1.3, which is installed on Ubuntu.	HTTP virtual services are exposed via hostnames are of the following form:
	`in-<unique_id>.<INGRESS_DOMAIN>`
	
	Note: The DNS used by a computer that runs a client application or the HTTP proxy that routes traffic to the virtual service needs to resolve such hostnames to the IP address of the ingress domain.
	
	You must perform the following actions that depend on the method you have used to set up the ingress domain:
	• If you used `nip.io` to form the ingress domain, then that virtual services host names are automatically resolved to the IP address of the ingress domain.
	For example, if you used `nip.io` to form the ingress domain as follows:
	`10.1.2.3.nip.io`
	The virtual services hostnames are resolved by `nip.io` to the IP address as `10.1.2.3`.
	• If your ingress domain does not use `nip.io` then you must ensure that the DNS in use by the client application or the HTTP proxy can resolve hostnames of the virtual services to the IP address of the ingress domain. For example, if the ingress domain is as follows:
	`myhost.mycom.com`
	The DNS must resolve `*.myhost.mycom.com` to the IP address of `myhost.mycom.com`.

HTTPS virtual service endpoints

When you want to run an HTTP virtual service where the HTTP transport settings have **Use SSL** enabled, the virtual service endpoint that is exposed for the virtual service instance expects an HTTPS request on port *443*. The exposed endpoint is

managed by the same ingress controller or the HAProxy that is used for the server endpoints. The SSL/TLS termination occurs at the endpoint before the request is routed to the virtual service instance and results in the following implications:

- The **Virtualization (Server) SSL** options that are specified in the HTTP transport have no effect.
- The certificate served by the exposed virtual service endpoint is the same as that of server that is signed by the server certificate authority.

Related information

Setting up intercepts on page 1457

Running HTTP virtual services in Kubernetes cluster without using proxies on page 1477

Setting up intercepts

When you want to run virtual services on Test , you can set up intercepts to route requests to the virtual service. You must configure the intercepts with the Test URL, team space ID, and offline user token that is generated from Test so that the intercepts can register with Test.

Examples of intercepts include the HTTP proxy component of HCL DevOps Test Virtualization Control Panel (Test Virtualization Control Panel) and the HCL DevOps Test Integrations and APIs (Test Integrations and APIs) Java virtualization agent available from the Test Virtualization Control Panel tools page.

Before you begin

You must have completed the following tasks:

- Installed the intercept from Test Virtualization Control Panel of the same version as that of Test. For example, installed the HTTP proxy.
- Read the following procedures:
  - Generating offline user tokens. See Access management for DevOps Test on page 1709.
  - Viewing the *Team Space ID* as the URL alias of the team space. See Viewing the configuration of a team space on page 1719.

About this task

You must edit the `registration.xml` file for the intercept to allow it to register with Test.

1. Perform the following steps in Test:
   a. Log in to Test.

      The **Team Space Dashboard** page is displayed.

   b. Generate the offline user token by clicking the **User** icon , and then click **Create Token**.

   c. Copy the offline user token to a text file.

d. Click the **Settings** icon of the team space in the **My Team Spaces** page.

The **Team Space Configuration** page is displayed.

e. Copy the text that is displayed under **URL alias** as the *Team Space ID*.

2. Perform the following steps to edit the `registration.xml` file for the intercept. For most intercepts this is in the root directory of the extracted intercept compressed file. For the HTTP/TCT proxy it is in the `<DTVCP_install_directory>\httptcp` directory on the computer on which you have installed Test Virtualization Control Panel.

a. Open the `registration.xml` file by using a text editor.

b. Enter the values for the following attributes:

Note: If an entry exists, you must either comment out that entry and add a new one or replace the contents with the values indicated.

Attribute	Value
server base-url	URL of Test
security-token	Offline user token generated from Test
team-space-id	ID of the team space to which the proxies must register

c. Save and close the file.

d. Perform any of the following actions:

- Restart the HTTP proxy, if you updated the `registration.xml` file of the HTTP/TCP proxy.
- Restart the application that is configured to use the any of the other intercepts other than HTTP/TCP.

3. Perform the following steps in Test to view the HTTP proxies that are registered in the team space:

a. Log in to Test.

b. Click **Infrastructure > Agents and Intercepts** to view the registered intercepts.

The intercepts that you configured with your offline user token and *Team Space ID* are displayed.

Results

You have successfully configured intercepts and viewed that they are registered with the team space in Test.

What to do next

You can perform the following tasks:

- Read for the prerequisite tasks that you must perform before you can run the HTTP virtual services. See Prerequisites for running HTTP virtual services in a Kubernetes cluster on page 1455.
- Run the virtual service resource. See Configuring a run of a virtual service on page 1462.
- View the routing rules for the intercept. See Viewing routing rules of the virtual services on page 1494.
- View the routing rules for the intercept in a running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 1487.

Viewing intercepts that are registered with a team space on DevOps Test

After you install the HTTP proxies and configure them with your offline user token, the proxies register with Test in a team space as intercepts and you can view the registered intercepts on the **Agents and Intercepts** page of the team space.

Before you begin

You must have completed the following tasks:

- Completed the prerequisite tasks before you configure a run of the HTTP virtual services. See Prerequisites for running HTTP virtual services in a Kubernetes cluster on page 1455.
- Installed the HTTP proxies of the same version as the version of Test.
- Configured the HTTP proxy with your offline user token so that the proxy can register as an intercept with Test. See Setting up intercepts on page 1457.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the API agents, intercepts, Kubernetes clusters, performance agents, or DevOps Test Runtime agents that are registered with Test.
3. You can view the intercepts that you own in any of the following ways:
   - Search for the intercept by entering the name of the intercept in the **Search** field.

     Note: You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.

   - Sort the **Type** column to show the items sorted with the intercepts in the rows at the top of the table.
   - Sort the **Agents** column and identify the intercepts by their names or by their owner.

   You can view the following details about the intercepts that are registered with Test:

- `Any project` is displayed in the **Projects** column indicating that the intercept can be used by virtual services in any project.
- The status of the intercept is displayed in the **Status** column.

Results

You have viewed the registered intercepts from the **Agents and Intercepts** page.

What to do next

You can start the virtual services that use the registered intercepts. See Configuring a run of a virtual service on page 1462.

Viewing virtual service resources

When project repositories contain virtual service resources and you want to view the resources so that you can start them, you can view the virtual services from the **Resources** page in your team space on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created virtual services that use any of the supported transports for tests in Test Integrations and APIs, committed the test resources to the remote repository, and added the repository to your project.

About this task

After you have committed the test assets and resources to the remote repositories and added the repositories to projects in your team space on Test, you can go to the **Resources** page and perform the following tasks:

- View the virtual service resources that are contained in any branch of the repositories that are added to your project.
- View the components in the System Model that are associated with the virtual service resources.
- Create or use saved filters for viewing virtual service resources.
- Run or start an instance of the virtual service resource.
- View the number of instances of the virtual service resources that are running.
- View all instances of a running virtual service on the **Instances** page by clicking the **Show in instances page** icon .

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Resources** in the navigation pane.

The **Resources** page is displayed.

4. Complete the steps for the task that you want to perform as listed in the following table:

Task	Action
Viewing the virtual service resources that are contained in a branch of any of the repositories that are added to your project.	Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field. All virtual services in the selected branch are displayed on the **Resources** page.
Viewing the components that are associated with the virtual service resource.	The components that are associated with the virtual service resource are displayed in the **Components** column. Click the component in the **Components** column to view the virtual resource on the **System Model** page.
Creating or using saved filters for viewing specific instances of virtual service resources.	To create a query, perform the following actions: a. Click **New filter**. b. Create a rule with an appropriate operator. c. Select criteria such as *Name*, *Path*, *Instance number*, *Instance state*, or *Components*. d. Select the condition, and then enter the value for the criteria. e. Apply the filter query. The virtual services that match the filter criteria are displayed. f. Save the filter query for retrieving it from the saved filters list. To use a saved filter query, perform the following actions: **Note:** To open the filter query, you must have created and saved a filter query. a. Click the **Open filters** icon b. Select the saved filter. c. Apply the filter. The virtual services that match the filter criteria are displayed.
Running an instance of the virtual service resource.	After you identify the virtual service that you want to run, perform the following actions:

Task	Action
	a. Click the **Execute** icon in the row of the identified virtual service. The **Execute virtual service** dialog is displayed. b. Run the virtual service with the settings that are configured in the resource by clicking **Execute** or modify any of the settings, and then click **Execute**. An instance of the virtual service starts to run. The virtual service is displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column.
Viewing the number of instances of the virtual service resources that are running.	If any of the virtual services in the project repositories are started by any project member with the *Tester* or *Owner* role, you can view the number of instances that are running or active from the **Resources** page. The virtual services are displayed with the state as `Running` along with the number of instances that are running, in the **Active instances** column.
Viewing all instances of a running virtual service.	Click the **Show in instances page** icon in the row of the virtual service on the **Resources** page. The **Instances** page is displayed with all the running instances of a particular virtual service. You can also view the number of requests that are received by the virtual service. The number of requests is displayed as the number of hits when you hover the cursor over the text in the **Activity** column.

Results

You have viewed virtual services from the **Resources** page on Test.

Related information

Configuring a run of a virtual service on page 1462

Configuring a run of a virtual service

When you have virtual service resources that are in the repositories added to your project on Test , you must configure a run of the virtual service to start an instance of the virtual service in the Test cluster, a remote Kubernetes cluster, or an API agent.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created virtual services that use any of the supported transports for tests in Test Integrations and APIs, committed the test resources to the remote repository, and added the repository to your project.
- Read about the virtual services that are supported on Test. See Supported virtual services on page 707.
- Completed the prerequisite tasks before you configure a run of the HTTP virtual services. See Prerequisites for running HTTP virtual services in a Kubernetes cluster on page 1455.
- Read and completed the tasks mentioned in Considerations for using a remote Kubernetes cluster on page 1101, if you want to use a remote Kubernetes cluster as a location to run the test.
- Read and completed the tasks mentioned in Considerations for running virtual services on remote API agents on page 1081, if you want to use a remote API agent as the location to run the virtual services.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Resources** in the navigation pane.

   The **Resources** page is displayed.
4. Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field.

   All virtual services in the selected branch are displayed on the **Resources** page.
5. Identify the virtual service by completing any of the following steps:
   - Search for the virtual service by entering the name or the path of the virtual service in the repository in the **Search** field box.
   - Create a filter query by using the **New filter** option and complete the following steps:
     a. Create a rule with an appropriate operator.
     b. Select criteria such as *Name*, *Path*, *Instance number*, or *Instance state*. Select the condition and enter the value for the criteria. Apply the filter query.

        The virtual services that match the filter criteria are displayed.
     c. Save the filter query for retrieving it from the saved filters list.
   - Retrieve a saved filter by using the **Open filters** icon by completing the following steps:

Note: To open the filter query, you must have created and saved a filter query.

a. Select the saved filter.
b. Apply the filter.

The virtual services that match the filter criteria in the filter that is applied are displayed.

You have identified the virtual service resources that you want to run.

6. Click the **Open action menu** icon in the row of the identified virtual service, and then click **Execute** icon.

The **Execute virtual service** dialog is displayed.

7. Select the version of the virtual service that is in the repository that you want to start by performing any of the following actions:

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:
  - Commit message.
  - Tags labeled for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version committed followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the environment that was used to bind the physical and logical resource in the API project, in the **ENVIRONMENT** tab.

Important: The configuration that you set for the run in the **Execute virtual service** dialog is preserved when you run the same virtual service again. The configurations that you set are not available to other members when they want to run the virtual service. For example, if you selected an environment, the same environment is selected when you run the virtual service again.

9. Enter a label, if required.

A label that you enter for the test run that helps you to identify the virtual service instance on the **Instances** page. The label that you entered is displayed for the virtual service under the **Labels** column on the **Instances** page. After you have created a label, any member of the project can use that label.

10. Follow the instructions if you want to modify the configurations for the behavior of the virtual service:

a. Click the **BEHAVIOR** tab, if it is not already open.

b. Configure or change the settings for the following options:

Note: The settings displayed are the settings that were configured for the virtual service when it was authored in Test Integrations and APIs.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Performance</td><td>The following settings are available for handling of requests by the virtual service:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Optimize performance</td><td>If enabled, attempts are made to reduce the amount of processing between the time the virtual service receives a request and the time it sends a response. Specific optimizations depend on the message contents, as in the following examples:
When the virtual service receives requests, all validations are disabled, and for all XML payloads, the store and filter actions are converted to use XPath expressions.
When the virtualization sends responses, any store actions set on a message are disabled, and any XML content is collapsed when the virtual service is compiled instead of being collapsed every time that a response is sent.
The optimization is disabled as the default action for this setting and you can enable the performance optimization if you fully understand the implications of optimizing the performance.</td></tr>
<tr><td>Threads</td><td>The maximum number of threads used in processing requests received by the virtual service. The default number of threads is 10.</td></tr>
</table></td></tr>
<tr><td>Operation</td><td>Specifies an operation referenced by the virtual service.</td></tr>
<tr><td>Response time</td><td>Specifies the response time behavior for responses sent by the virtual service for the selected operation.</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>You can modify the value by selecting a value from the following options:
<table><tr><th>Option</th><th>Description</th></tr><tr><td>No delay</td><td>Select this option for a response with no delay.</td></tr><tr><td>Minimum delay</td><td>Select this option for a delay in the response by entering the required delay in milliseconds (ms).</td></tr><tr><td>Uniform distribution</td><td>Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms).</td></tr><tr><td>Gaussian distribution</td><td>Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms).</td></tr></table></td></tr>
<tr><td>Passthrough</td><td>Specifies the pass through behavior for the selected operation when requests are not handled within the virtual service.

You can modify the value by selecting a value from the following options:
<table><tr><td>Discard</td><td>This option stops the system under test from receiving the intercepted message. This option can disrupt the calling system. For example, the system might time out while it waits for a reply.</td></tr><tr><td>Pass Through</td><td>This option passes the intercepted message to the system under test, with an optional delay.</td></tr><tr><td>Simulate Error</td><td>This option returns an error to the calling system. The message is not passed to the system under test.</td></tr></table></td></tr>
</table>

11. Follow the instructions, if the virtual service references datasets.
    - You can use the dataset referenced in the asset.
    - You can choose to override the dataset with another data source. If alternative data sources are available, select from the set of overrides available.
12. Follow the instructions if the virtual service requires a variable that must be passed at run time.
    a. Click the **VARIABLES** tab, if it is not already open.
    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

Note: You must have created a file with the variables before you can select the file.

13. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed at run time in the **JVM Arguments** field.

    Note: Each JVM argument should be separated with white space.

    For example, you can set a maximum Java heap size.

    b. Enter the environment variables that must be passed at run time in the **Environment Variables** field, if applicable.

    For example, enter the environment variables when the third-party libraries that are used in the run refer to the environment variables for configuration.

    c. Select the stub logging level for the virtual service from the following options in the **Logging** list:

Option	Description
None	Specifies that the virtual service does not write log messages.
Normal	Specifies that the virtual service writes informational messages.
Debug	Specifies that the virtual service writes informational and debugging messages.

    d. Enter other configuration options as parameters and their values in the **Additional Configuration Parameters** fields, if applicable.

    You can refer to the additional parameters that you can use for virtual services from the topic in the related links.

    For example, if you want to start the virtual service to run in a new container, you can specify the following parameter and its value:

Parameter name	Value
stub.dedicated.container	**true**

Note: Click **Add** to add additional parameters.

Note: You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for the fields for the advanced settings is null or an empty field.

Notes:

- If you have configured some or all of the settings for the current run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

14. Select the step for the location to run the virtual services based on the following types of virtual services:
    - If you want to run any of the following types of virtual services only in the default Kubernetes cluster or a remote Kubernetes cluster, then go to Step 15 on page 1468.
    - If you want to run the virtual services on a remote API agent that you added to your project, then go to Step 16 on page 1469.
15. Follow the instructions if you want to change the location of a Kubernetes cluster for running the test:
    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the virtual service resource runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    Important: You must have added the registered remote Kubernetes clusters to your project that are then displayed under the **Override** column:

    Notes:

- If the remote clusters are not added to your project, the option **No override options** is displayed as the default value and the virtual service resource runs in the Kubernetes cluster of Test.
- If remote clusters are added to your project, the added clusters are displayed along with their availability status and ownership information.

b. Select the location where you want to run the virtual service resource from the following options:

- Select the **Default Cluster** when no remote clusters are available in your project.
- Select the remote cluster from the list when a remote cluster is available in your project.
- Select **No override options**, if you selected any remote cluster and want to revert to the **Default Cluster** to run the virtual service resource.

c. Click **Execute**.
The run is initiated.

16. Follow the instructions if you want to run the virtual services on the remote API agent that is connected to the default Kubernetes cluster:

a. Click the **LOCATION** tab, if it is not already open.

The API agents from the list of registered API agents that match with the attributes or tags configured for the virtual service resources in the project are listed under the **Host** column. The information about the availability of the agent is displayed.

Note: You must have added the API agents to your project from the **Agents and Intercepts** page for the API agents to be displayed under the **Override** column.

The default value for the API agents is null or an empty field if no agents were configured in the project resources. If the project resources contains API agents that match with the agent tags, then the default API agent is the first item to be displayed on the list of API agents listed in the increasing alphabetical order.

b. Select the API agent where you want to run the virtual service in the following scenarios:

If...	**Then...**	**Action**
The API agent that is specified in the project resources is available and no other agents are added to the project.	The API agent that is specified in the project resources is displayed in the **Override** column.	Select the API agent in the **Host** that is also displayed in the **Override** column as the location to run the test.
The API agent that is specified in the project resources is not available and no other API agents are added to the project.	There is no API agent displayed for override in the **Override** column.	You cannot run the test on the API agent.

If...	Then...	Action
		You must wait until the API agent is available to run the test on the API agent.
The API agent that is specified in the project resources is available and other API agents added to the project.	API agents that have capabilities that match with the API agent capabilities specified in the project resources are listed in the **Override** column as follows: ▪ API agents with capabilities that best match the capabilities of the API agent in the project resources are at the top of the list. ▪ API agents with capabilities that do not match with the capabilities of the API agent in the project resources are listed subsequently. You can find details of both API agents for the matching capabilities when you hover over the agent in the **Override** column.	Perform any of the following actions: ▪ Select the API agent in the **Host** that is also displayed in the **Override** column as the location to run the test. ▪ Select an API agent in the **Override** column as the location to run the test.

c. Click **Execute**.

The test run is initiated.

Results

You have started a virtual service from the **Resources** page on Test.

What to do next

After you start a run of a virtual service, you can view the status of the virtual service in any of the following ways:

- On the **Resources** page, the virtual services are displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column.
- On the **Resources** page, you must click the **Show in instances page** icon in the row of the virtual service to view all the running instances of a particular virtual service on the **Instances** page.

- Go to the **Instances** page by clicking **Virtualization > Instances**. You can view instances of all the virtual services that are running and are displayed with the state as `Running` under the **State** column. You can also view the requests that are received by the virtual service. The number of requests is displayed as number of hits when you hover the cursor over the text in the **Activity** column.

You can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 1487.

You can modify the behavior or the logging level of a running instance of the virtual service that you started before running tests on them. See Modifying configurations of running instances of virtual services on page 1492.

When you have completed testing the virtual service, you must stop the running virtual service. See Stopping virtual services on page 1498.

Related reference

Additional configuration parameters for virtual services on page 1471

Additional configuration parameters for virtual services

You can find information about the additional configuration parameters that you can set when you want to run virtual services on Test.

When you use the server UI to start the virtual service, some of the parameters are handled by fields in the **Execute virtual service** dialog.

You can enter the other parameters as a *name-value* pair in the **Additional configuration parameters** field that is displayed in the **Advanced** settings panel of the **Execute virtual service** dialog.

The following additional configuration parameters are supported when you configure a run for a virtual service.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
To specify a unique identifier for use in the generated host name. **Note:** This parameter applies only if the virtual service is run in a Kubernetes	`stub.ingress.host.identifier`	Any <custom_id> that you specify. **Note:** The **<custom_id>** must start and end with an alphanumeric character. The characters supported	*mystub*	See Running HTTP virtual services in Kubernetes cluster without using proxies on page 1477, to know how the identifier you specified is used.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
cluster and not on an API agent.		include a-z, 0-9, and a hyphen. No other characters can be used in the **<custom_id>**. The **<custom_id>** must not be longer than 32 characters.		
To specify whether a virtual service instance should be run in a dedicated container. **Note:** This parameter applies only if the virtual service is run in a Kubernetes cluster and not on an API agent.	`stub.dedicated.container`	• *true* • *false* **Note:** The default value is *false*.	*true*	A dedicated container is used to run a single virtual service instance.
To specify whether performance optimization is enabled	`stub.performance.optimize`	• *true* • *false*	*true*	The performance optimization is enabled for the virtual service.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
for the virtual service instance. The value set overrides the setting from the stub definition. This option can be set in the **Behavior** tab from the UI.		**Note:** The default value is *false*.		
To specify the number of threads to be used for processing requests for the virtual service instance. The value set overrides the setting from the stub definition. This option can be set in the **Behavior** tab from the UI.	`stub.worker.thread.count`	Any number that you specify as number of threads to use.	*15*	15 threads are used for processing requests for the virtual service instance.
To specify the stub logging level for the virtual service instance. The value set overrides the setting from the stub definition. This option can be set in the **Advanced** settings from the UI.	`stub.logging.level`	• *none* • *normal* • *debug*	*debug*	The virtual service writes informational and debugging messages to the container log.
To specify the pass-through behaviour type of the virtual service instance. The value set overrides the	`stub.pass.through.behavior`	Valid values depend on the stub operation transport type but can be as follows:	*simulate_error*	This option returns an error to the calling system. The message is not passed to the system under test.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
setting from the stub definition. This option can be set in the **Behavior** tab from the UI.		• *simulate_error* • *discard* • *pass_through*		
To specify the delay in passing through from the virtual service instance. The value set overrides the setting from the stub definition.	`stub.pass.through.delay.period`	Any number of seconds in milliseconds.	*10000*	A delay of 10 seconds in passing through from the virtual service instance.
To specify the status code to use in the virtual service instance response when simulating an error. The value set overrides the setting from the stub definition. **Note:** Applies to HTTP stubs. This option can be set in the **Behavior** tab from the UI.	`stub.pass.through.status.code`	You must specify a valid HTTP status code as the value. For example, `503`.	*500*	The HTTP error code of `500` is sent in the response if the virtual service simulates an error when passing through.
To specify the reason phrase to use in the virtual service instance response when simulating an error. The value set overrides the setting from the stub definition.	`stub.pass.through.reason.phrase`	You must specify the text of the status message as the value.	*my custom error msg*	The text that you specify is set as the HTTP reason phrase in the response if the virtual service simulates an error when passing through.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
Note: Applies to HTTP stubs. This option can be set in the **Behavior** tab from the UI.				
To specify the type of response-time behavior that is required for the virtual service instance. The value set overrides the setting from the stub definition. This option can be set in the **Behavior** tab from the UI.	`stub.response.time.type`	• *no_delay* • *minimum_delay* • *gaussian* • *uniform* • *performance_profile* **Note:** *performance_profile* is only valid if the stub operation in the stub definition is defined to use a performance profile.	*no_delay*	The virtual service sends a response without any delay.
To specify the minimum time for the response time behaviour required for the virtual service instance. The value set overrides the setting from the stub definition.	`stub.response.time.minimum`	Any number of seconds in milliseconds.	*5000*	The virtual service sends a response after a minimum delay of 5 seconds.

Requirement	Configuration parameter name	Supported values	An example value	Result of using the example value
This option can be set in the **Behavior** tab from the UI.				
To specify the maximum time for the response time behaviour required for the virtual service instance. The value set overrides the setting from the stub definition. This option can be set in the **Behavior** tab from the UI.	`stub.response.time.maximum`	Any number of seconds in milliseconds.	*20000*	The virtual service must send a response after a maximum delay of 20 seconds.
To override the value of for an input tag used by the virtual service.	`stub.input.tag.`*`name`*, where *name* is the name of the input tag that you want the virtual service to use.	The value to set for the input tag.	You can specify a tag name as `stub.input.tag.mytag` with a value of *red*.	The input tag named *mytag* will have the value *red* when the virtual service is run.

Virtual services where the stub definition refers to multiple operations

When a stub definition has two or more events that refer to different operations and each operation has a different behavior, you must specify the *<operation_id>* suffixed to the configuration parameter name in the **Additional configuration parameters** field.

Note: The *<operation_id>* is the internal ID of the operation that is found in the **Documentation** tab when you open the operation in Test Integrations and APIs.

When the stub definition refers to multiple operations and you want to start a run of a virtual service, you must append the *<operation_id>* to the *<parameter_name>* in the **Additional configuration parameters** field.

The configuration parameter names that you must use for the virtual service, which is associated with a particular operation are as follows:

- `stub.pass.through.behavior.<operation_id>`
- `stub.pass.through.delay.period.<operation_id>`

- `stub.pass.through.status.code.<operation_id>`
- `stub.pass.through.reason.phrase.<operation_id>`
- `stub.response.time.type.<operation_id>`
- `stub.response.time.<operation_id>`
- `stub.response.time.maximum.<operation_id>`

For example, if the *<operation_id>* is *7281b750:162483a09f1:-7e68*, the parameter name for the `stub.response.time.type` option that you must specify is `stub.response.time.type.7281b750:162483a09f1:-7e68`.

Related information

Configuring a run of a virtual service on page 1462

Running HTTP virtual services in Kubernetes cluster without using proxies

When you run HTTP virtual services in the Kubernetes cluster of Test or a remote Kubernetes cluster, the endpoint on which the virtual service is exposed externally uses a host name that includes a generated ID. If a client application is required to call the virtual service directly via its external hostname and not via the HTTP proxy, then you might want to determine the hostname to configure the client first before you start the virtual service.

Important: This task does not apply when you use an API agent to run the HTTP virtual services.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in a team space on Test and added the repository that contains the HTTP virtual services.
- Completed the prerequisite tasks before you start HTTP virtual services. See Prerequisites for running HTTP virtual services in a Kubernetes cluster on page 1455.
- Selected the branch of the Git repository to view test resources on the **Execution** page.

About this task

When you installed Test on Ubuntu, the default HTTP virtual service endpoints are of the following form:

```
in-<unique_id>.<ingress_domain>
```

For example, with an ingress domain as **10.1.2.3.nip.io**, a virtual service endpoint might be as follows:

```
in-cd3a755635574c6fa7e264911301821a.10.1.2.3.nip.io
```

When you installed Test 10.2.1 or earlier on Red Hat OpenShift, the default HTTP virtual service endpoints are of the following form:

```
<ingress_domain_first_label>-<unique_id>.<remainder_of_ingress_domain>
```

For example, with an ingress domain as **dth.mycluster-123456-0000.region.containers.appdomain.cloud**, a virtual service endpoint might be as follows:

```
dth-cd3a755635574c6fa7e264911301821a.mycluster-123456-0000.region.containers.appdomain.cloud
```

The **<unique_id>** is generated when the virtual service is started. The full hostname can be viewed in the **Routing to** field of the routing rule created for the virtual service on the **Intercept Rules** page. When traffic is routed to the virtual service via the HTTP proxy, a client application is unaware of this hostname. If the client application is required to call the virtual services directly, it must be aware of the hostname of the virtual service.

To allow a hostname to be known before you start the virtual service, you can replace the generated ID with a user defined value that is specified as an additional configuration parameter when starting the stub. This value is referred to as **<custom_id>**.

Notes:

- The value of the **<custom_id>** that you specify is used as the **<unique_id>** for the virtual service.
- The **<custom_id>** must start and end with an alphanumeric character. The characters supported include a-z, 0-9, and a hyphen. No other characters can be used in the **<custom_id>**. The **<custom_id>** must not be longer than 32 characters.

Restriction: The **<custom_id>** must be unique to a virtual service that is running. You must not assign the same **<custom_id>** to another virtual service that is running simultaneously. Doing so might result in an undefined behavior of the virtual service.

For example, on Ubuntu, if you specify the value of the **<custom_id>** for the virtual service as `stub123-test` and Test has an ingress domain as **10.1.2.3.nip.io**. Then the hostname of the virtual service endpoint is as follows:

```
in-stub123-test.10.1.2.3.nip.io
```

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Resources** in the navigation pane.

   The **Resources** page is displayed.

4. Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field.

   All virtual services in the selected branch are displayed on the **Resources** page.

5. Identify the virtual service that you want to run.
6. Click the **Open action menu** icon in the row of the identified virtual service, and then click **Execute** icon .

   The **Execute virtual service** dialog is displayed.

7. Select the version of the virtual service that is in the repository that you want to start by performing any of the following actions:

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version committed followed by the versions committed previously.

   - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

     The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the environment that was used to bind the physical and logical resource in the API project, in the **ENVIRONMENT** tab.

   Important: The configuration that you set for the run in the **Execute virtual service** dialog is preserved when you run the same virtual service again. The configurations that you set are not available to other members when they want to run the virtual service. For example, if you selected an environment, the same environment is selected when you run the virtual service again.

9. Enter a label, if required.

   A label that you enter for the test run that helps you to identify the virtual service instance on the **Instances** page. The label that you entered is displayed for the virtual service under the **Labels** column on the **Instances** page. After you have created a label, any member of the project can use that label.

10. Follow the instructions if you want to modify the configurations for the behavior of the virtual service:

    a. Click the **BEHAVIOR** tab, if it is not already open.

    b. Configure or change the settings for the following options:

Note: The settings displayed are the settings that were configured for the virtual service when it was authored in Test Integrations and APIs.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Performance</td><td>The following settings are available for handling of requests by the virtual service:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Optimize performance</td><td>If enabled, attempts are made to reduce the amount of processing between the time the virtual service receives a request and the time it sends a response. Specific optimizations depend on the message contents, as in the following examples:
When the virtual service receives requests, all validations are disabled, and for all XML payloads, the store and filter actions are converted to use XPath expressions.
When the virtualization sends responses, any store actions set on a message are disabled, and any XML content is collapsed when the virtual service is compiled instead of being collapsed every time that a response is sent.
The optimization is disabled as the default action for this setting and you can enable the performance optimization if you fully understand the implications of optimizing the performance.</td></tr>
<tr><td>Threads</td><td>The maximum number of threads used in processing requests received by the virtual service. The default number of threads is 10.</td></tr>
</table></td></tr>
<tr><td>Operation</td><td>Specifies an operation referenced by the virtual service.</td></tr>
<tr><td>Response time</td><td>Specifies the response time behavior for responses sent by the virtual service for the selected operation.

You can modify the value by selecting a value from the following options:</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>No delay</td><td>Select this option for a response with no delay.</td></tr>
<tr><td>Minimum delay</td><td>Select this option for a delay in the response by entering the required delay in milliseconds (ms).</td></tr>
<tr><td>Uniform distribution</td><td>Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms).</td></tr>
<tr><td>Gaussian distribution</td><td>Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms).</td></tr>
</table>
</td></tr>
<tr><td>Passthrough</td><td>Specifies the pass through behavior for the selected operation when requests are not handled within the virtual service.

You can modify the value by selecting a value from the following options:
<table>
<tr><td>Discard</td><td>This option stops the system under test from receiving the intercepted message. This option can disrupt the calling system. For example, the system might time out while it waits for a reply.</td></tr>
<tr><td>Pass Through</td><td>This option passes the intercepted message to the system under test, with an optional delay.</td></tr>
<tr><td>Simulate Error</td><td>This option returns an error to the calling system. The message is not passed to the system under test.</td></tr>
</table>
</td></tr>
</table>

11. Click **Advanced** to make the following advanced configurations:

    a. Enter the following configuration options as parameters and their values in the **Additional Configuration Parameters** fields.

Parameter name	**Value**
stub.ingress.host.identifier	**<custom_id>**

12. Click **Execute**.

    The run is initiated.

Results

You have started an HTTP virtual service that does not use a proxy. The virtual service exposes the hostname that contains a custom ID that you specified.

What to do next

After you start a run of a virtual service, you can view the status of the virtual service in any of the following ways:

- On the **Resources** page, the virtual services are displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column.
- On the **Resources** page, you must click the **Show in instances page** icon in the row of the virtual service to view all the running instances of a particular virtual service on the **Instances** page.
- Go to the **Instances** page by clicking **Virtualization > Instances**. You can view instances of all the virtual services that are running and are displayed with the state as `Running` under the **State** column. You can also view the requests that are received by the virtual service. The number of requests is displayed as number of hits when you hover the cursor over the text in the **Activity** column.

Viewing running instances of virtual services

After you start virtual services from the **Resources** page, you can view the instances of the virtual services that are running from the **Instances** page on Test . You can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service. You can also stop the running instance.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

About this task

After you started the virtual services from the **Resources** page on Test, you can go to the **Instances** page and you can perform the following tasks:

- View all running instances of the virtual services that are contained in the project repositories.
- View the components in the system model that are associated with the virtual service instances.
- Search for a specific instance of the virtual service for which you want to view the details.
- Create or use saved filters for viewing specific running instances of the virtual service.
- View the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service.
- Change the behavior or logging level of a running instance of the virtual service before you run tests on the virtual service.

- Stop a specific running instance of the virtual service.
- Stop multiple instances or all instances of the virtual service that are running.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed.

   Any one of the following views is displayed:
   - No instances of virtual services are displayed if there are no instances that are running or were stopped in the past hour.
   - All running instances of virtual services in your project are displayed.
   - No running instances of virtual services are displayed but the option **Show all** is enabled. You can click **Show all** to display the instances that were stopped in the past hour. Instances that were hidden from the display when **Hide inactive** was clicked are displayed.

     Note: All instances that were stopped are automatically removed from the display after *60 minutes* of being stopped. You cannot use the **Instances** page to view the history of stopped instances. You can hide the stopped instances from the display by clicking **Hide inactive**, so that you can view only the running instances.

   You can find the following information about the running instances that are displayed on the **Instances** page.
   - The name and path of the virtual service in the project repository are displayed in the **Name** column.
   - The tags or labels added to the instance when the run was configured is displayed in the **Labels** column.
   - The name of the environment associated with the operation in which the virtual service was created in Test Integrations and APIs is displayed in the **Environment** column.
   - The state of the running instance is displayed as `Running` in the **State** column.
   - The activity of the running instance of the virtual service based on the number of requests received in the previous hour is displayed in the **Activity** column. The activities are classified and displayed as indicated in the following table:

Activity load	**Activity displayed as**
Received *0* requests	`None`
Received *1 - 10* requests	`Low`
Received *11 - 100* requests	`Mid`
Received *101 - 1000* requests	`High`

Activity load	Activity displayed as
Received more than *1000* requests	`Very High`

- The **Stop** icon is displayed in the **Actions** column and stops the running instance of the virtual service, when clicked.

4. Complete the steps for the task that you want to perform as listed in the following table:

Task	Action
Searching for a specific instance of the virtual service for which you want to view the details or want to stop the instance.	Enter either the complete text or part of the text in the name or the path of the virtual service that is in the repository in the **Search** field. The instances of the virtual services that match the text searched are displayed.
Viewing the component associated with the running instance of the virtual service.	You can view the name of the associated component that is displayed on the **Instances** page in the **Filter** row only if you clicked the icon on the **System Model** page to view all running instances of the virtual services .
Creating or using saved filters for viewing specific instances of virtual service resources.	To create a query, perform the following actions: a. Click **New filter**. b. Create a rule with an appropriate operator. c. Select criteria such as *Name*, *Path*, *Labels*, *Environment*, *State*, *Activity*, or *Location*. d. Select the condition, and then enter the value for the criteria. e. Apply the filter query. The virtual services that match the filter criteria are displayed. f. Save the filter query for retrieving it from the saved filters list. To use a saved filter query, perform the following actions: **Note:** To open the filter query, you must have created and saved a filter query. a. Click the **Open filters** icon. b. Select the saved filter. c. Apply the filter. The virtual services that match the filter criteria are displayed.
Viewing the details, configuration settings, routing rules, usage	After you identify the virtual service instance for which you want to view the details, perform the following actions:

Task	Action
statistics, or logs of a specific running instance of the virtual service.	a. Click the **Expand** icon in the row of the identified instance of the virtual service. The **Details** panel of the instance is displayed. You can find the following details about the instance displayed: ▪ Name ▪ Path ▪ Labels ▪ API Environment ▪ Asset ID ▪ Git Repo ▪ Version ▪ Started at ▪ Started by ▪ Execution location ▪ Associated components ▪ Execution location b. Click **Configuration** to expand the panel and view the configuration details of the virtual service. You can find the following configuration details about the instance displayed: ▪ Dataset overrides ▪ Variables ▪ JVM args ▪ Environment variables ▪ Additional parameters c. Click **Behavior** to expand the panel and view the behavior that was configured for the virtual service. d. Click **Routing Rules** to expand the panel and view the routing rules that were configured in the virtual service. e. Click **Statistics** to expand the panel and view the graphical representation of the activities performed by the virtual service. f. Click **Diagnostics** to expand the panel and view the logging configuration of the virtual service and the links to the logs.
Changing the behavior or logging level of a running instance of the virtual service.	After you identify the virtual service instance for which you want to view the details, perform the following actions: a. Click the **Expand** icon in the row of the identified instance of the virtual service. The **Details** panel of the instance is displayed.

Task	Action
	Perform any of the following actions: ◦ Click **Behavior** to expand the panel. Perform the following steps to change the behavior configurations: a. Click the **Edit** icon . b. Select a different operation for the virtual service, if available. c. Select a different response time and enter the values for the delay that you want to set. d. Click **apply**. ◦ Click **Diagnostics** to expand the panel. Perform the following steps to change the logging level configuration: a. Click the **Edit** icon . b. Select a different logging level. c. Click **apply**.
Stopping a specific running instance of the virtual service.	a. Click the **Stop** icon in the row of the virtual service instance. b. Click **Ok** in the **Stop virtual service instance** dialog that is displayed. The running virtual service instance is stopped.
Stopping multiple running instances or all the running instances of virtual services.	a. Perform any of the following actions: ▪ Select the checkbox in the row of the virtual service instances that you want to stop, and then click **Stop selected**. ▪ Select the **Select all** checkbox in the header row, and then click **Stop selected**. **Note:** All running instances are selected when you click the **Select all** checkbox. b. Click **Ok** in the **Stop all selected virtual service instances** dialog that is displayed. The running virtual service instances are stopped.

Results

You have viewed running instances of the virtual services from the **Instances** page on Test.

What to do next

You can perform the following tasks:

- View the configurations of a running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 1487.
- Modify the behavior or logging level of a running instance of the virtual service before you run tests on the virtual service. See Modifying configurations of running instances of virtual services on page 1492.
- Stop a running instance or all running instances of the virtual services. See Stopping virtual services on page 1498.

Viewing configurations of running instances of virtual services

After you start virtual services from the **Resources** page, you can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running virtual service instance from the **Instances** page on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

About this task

After you have started the virtual services from the **Resources** page on Test, you can go to the **Instances** page and view the information about a specific running virtual service instance.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed with all running instances of the virtual services that are in your project.
4. Identify the running instance of the virtual service of which you want to view the details, configuration settings, and other information.
5. Click the **Expand** icon of the instance of the virtual service.

   The **Details** panel of the instance is displayed with the following details about the virtual service instance:

Note: The information displayed pertains to the configuration settings that were configured in the virtual service at the time it was created or run.

Option	Information displayed
Name	Displays the name of the virtual service.
Path	Displays the path of the virtual service resource in the project repository.
Labels	Displays the labels or tags that were entered for the virtual service when a run of the virtual service was configured.
API Environment	Displays the name of the environment associated with the operation in which the virtual service was created in Test Integrations and APIs or the environment selected for the run.
Asset ID	The ID of the virtual service resource in the project repository that was started as an instance.
Git Repo	The git repository added to the project that contains the virtual service resources.
Version	Displays the version of the virtual service resource that is in the project repository that was started as an instance.
Started at	Displays the date and time when the virtual service instance started.
Started by	Displays the name of the project member who started the virtual service instance.
Associated components	Displays the components in the system model that are associated with the virtual service.
Execution location	Displays the location of the Kubernetes cluster on which the virtual service instance ran. **Note:** The name of the remote Kubernetes cluster is displayed if the virtual service instance ran on the remote cluster or *Default Cluster* is displayed if the virtual service ran on Test.

The other panels that you can view are as follows:

- Configuration
- Behavior
- Routing Rules
- Statistics
- Diagnostics

6. Expand each panel by clicking the panel to view the information in that panel.

   You can view the following information in the different panels:

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td>Configuration</td><td>
The Configuration panel of the instance is displayed when expanded with the following details about the virtual service instance:

Note: The information displayed pertains to the configuration settings that were configured in the virtual service at the time it was created or run.

<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Dataset overrides</td><td>Displays the dataset overrides that were configured when executing the virtual service instance.</td></tr>
<tr><td>Variables</td><td>Displays the variables that were configured when executing the virtual service instance.</td></tr>
<tr><td>JVM args</td><td>Displays the JVM arguments that were configured when executing the virtual service instance.</td></tr>
<tr><td>Environment variables</td><td>Displays the environment variables that were configured when executing the virtual service instance.</td></tr>
<tr><td>Additional parameters</td><td>Displays the additional parameters that were configured when executing the virtual service instance.</td></tr>
</table>
</td></tr>
<tr><td>Behavior</td><td>
The Behavior panel of the instance is displayed when expanded with the following details about the virtual service instance:

Note: The information displayed pertains to the configuration settings that were configured in the virtual service at the time it was created or run.

<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Performance</td><td>Displays the optimization setting and the maximum number of threads in use by the virtual service.</td></tr>
<tr><td>Operation</td><td>Displays the operations referenced by the virtual service. You can select each operation to view the related response time and the pass through settings.</td></tr>
<tr><td>Response time</td><td>Displays the response time option that is in use for the selected operation within the virtual service.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Passthrough</td><td>Displays the pass through option that is in use for the selected operation within the virtual service.</td></tr>
</table>
</td></tr>
<tr><td>Routing Rules</td><td>
The Routing Rules panel of the instance is displayed when expanded.

The routing rules that relate to the virtual service are displayed.

<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>Activity</td><td>Indicates the routing type.</td></tr>
<tr><td>Target</td><td>Displays the endpoint of the target service for which requests are being routed.</td></tr>
<tr><td>Recipient</td><td>Displays the endpoint of the virtual service that receives the routed requests</td></tr>
</table>

You can expand the panel for each intercept by clicking on the expand icon . You can then find the following information:

<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td>Details</td><td>
The Details panel provides the following information:

<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Condition</td><td>Lists the condition that causes the traffic to be routed to a virtual service.

You can view the detailed condition of the rule defined for the intercept by clicking the View icon .

The detailed condition of the rule is displayed in the Full condition window.

 Note: You can copy the detailed condition if you intend to use it elsewhere.</td></tr>
<tr><td>Routing to</td><td>Lists the virtual service to which traffic is routed.</td></tr>
<tr><td>Created by</td><td>Lists the user who created the rule.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Created at</td><td>Lists the date and time when the rule was created.</td></tr>
</table>
</td></tr>
<tr><td>Activity</td><td>The Activity panel lists the log of the activities performed in relation to the routing rule. This panel also displays the details of the activity, date and time, and message for that activity.</td></tr>
</table>
</td></tr>
<tr><td>Statistics</td><td>The Statistics panel of the instance is displayed when expanded.

The requests received by the instance of the virtual service in the previous hour are displayed as a graph. The total requests received is indicated in the Total hits field and the activity load in the previous hour are displayed in the Activity field.

The view can be refreshed by clicking the Refresh icon .</td></tr>
<tr><td>Diagnostics</td><td>The Diagnostics panel of the instance is displayed when expanded with the following details:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Logging</td><td>Displays the logging level that was configured for the virtual service.</td></tr>
<tr><td>Execution ID</td><td>Displays the ID of execution for the virtual service instance.</td></tr>
<tr><td>Stubs runtime ID</td><td>Displays the ID of the runtime where the virtual service instance runs. If executed on a cluster the value is that of the label execution-stubs-runtime-id on the pod that runs the virtual service instance. If executed on an API agent, the value is the ID of the engine on which the virtual service instance runs.</td></tr>
<tr><td>Init container log</td><td>Displays the link to the log of the Init container.</td></tr>
</table>
</td></tr>
</table>

Panel	Details	
	Option	**Description**
		Note: This logs contains messages that relate to configuring the container which is ready to run the virtual service.
	Main container log	Displays the link to the log of the main container. This log contains messages logged by the virtual service based on the selected logging level.
	Pod events log	Displays the link to the log of the pod events log. This log contains messages logged in the pod for the virtual service events.

Results

You have viewed the information about the configuration, routing rules, usage, or logs of the virtual service instance.

What to do next

You can modify the behavior or logging level configurations of the running virtual service instance before you run tests on the instance. See Modifying configurations of running instances of virtual services on page 1492.

Related information

Configuring a run of a virtual service on page 1462

Modifying configurations of running instances of virtual services

After you start virtual services from the **Resources** page and before you run tests on the virtual service, you might want to modify the configurations for the behavior or the logging level of the running instance.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

About this task

After you have started the virtual services from the **Resources** page on Test , you can go to the **Instances** page and identify the running instance of the virtual service for which you want to modify the configurations. You can modify the configurations for the behavior or logging levels of the running instance of the virtual service.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects >** ***project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed with all running instances of the virtual services that are in your project.
4. Identify the running instance of the virtual service for which you want to modify the configurations.
5. Click the **Expand** icon of the instance that you want to modify the configurations.

   The **Details** panel of the instance is displayed.
6. Click **Behavior** to expand the panel and perform the following steps to change the configurations:

   a. Click the **Edit** icon .

      The options that you can modify are enabled for editing.

   b. Select a different operation for the virtual service, if available, from the list for the **Operation** option.

   c. Select a different response time from the list for the **Response time** option, and then enter the values for the delay that you want to set.

      You can modify the value by selecting a value from the following options:

Option	Description
No delay	Select this option for a response with no delay.
Minimum delay	Select this option for a delay in the response by entering the required delay in milliseconds (ms).
Uniform distribution	Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms).
Gaussian distribution	Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms).

   d. Click **apply** to apply the modified settings.

7. Click **Diagnostics** to expand the panel and perform the following steps to change the logging level that is configured:

    a. Click the **Edit** icon .

    The **Logging level** option is enabled for editing.

    b. Select a different logging level from the following options.

Option	Description
None	Specifies that the virtual service does not write log messages.
Normal	Specifies that the virtual service writes informational messages.
Debug	Specifies that the virtual service writes informational and debugging messages.

    c. Click **apply** to apply the modified settings.

Results

You have viewed and modified the configurations of a running virtual service instance from the **Instances** page on Test.

What to do next

You can run tests that are in an API Suite in your project, which then run on the virtual service instance. See Configuring an API Suite or an API test run on page 1174.

Viewing routing rules of the virtual services

When you run virtual services that are created to run on proxies or intercepts and have routing rules defined, you can view the proxies or intercepts and the routing rules from the **Intercept Rules** page.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

About this task

After you run the virtual service resources that are in your project on Test , you can view the routing rules for the running instances of the virtual service resources, from the **Intercept Rules** page.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

The **Overview** page is displayed.

3. Click **Virtualization > Intercept Rules** in the navigation pane.

   The **Intercept Rules** page is displayed.

   The agents, proxies, or intercepts that are configured in the virtual service resources that are in your project are displayed.

   You can find the following information that is displayed on the **Intercept Rules** page.

Column	Description
Activity	Lists the agent or proxy.
Target	Lists a summary view of the endpoint that was recorded or virtualized.
Recipient	Lists the destination where messages sent to are captured.

4. Expand the proxy or intercept panel by clicking the **Expand** icon .

   The details of the proxy or intercept are displayed in the **Details** tab in the expanded panel.

5. Click each tab to view the following information:

<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td>Details</td><td>The Details panel provides the following information:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Condition</td><td>Lists the condition that causes the traffic to be recorded or routed to a virtual service.

You can view the detailed condition of the rule defined for the proxy or intercept by clicking the View icon .

The detailed condition of the rule is displayed in the Full condition window.

Note: You can copy the detailed condition if you intend to use it elsewhere.</td></tr>
<tr><td>Routing to</td><td>Lists the virtual service to which traffic is routed.</td></tr>
<tr><td>Created by</td><td>Lists the user who created the rule.</td></tr>
<tr><td>Created at</td><td>Lists the date and time when the rule was created.</td></tr>
</table></td></tr>
<tr><td>Activity</td><td>The Activity panel lists the log of the activities performed by the proxy or intercept. This panel also displays the details of the activity, date and time, and message for that activity.</td></tr>
</table>

Results

You have viewed the routing rules for the virtual service, proxy, or intercept that are running or connected to Test.

What to do next

You can perform any of the following tasks:

- You can view the usage statistics of the virtual services that are running. See Viewing usage statistics of virtual services on page 1496.
- You can stop the running instances of the virtual services. See Stopping virtual services on page 1498.

Related information

Management of virtualized services on page 1453

Viewing usage statistics of virtual services

After you start virtual service resources that are in the project repositories, and then run tests on the virtual service, you can view the usage statistics of the virtual services from the **Usage** page on Test .

Before you begin

You must have completed the following tasks:

- Completed the prerequisite tasks before you configure a run of the HTTP virtual services. See Prerequisites for running HTTP virtual services in a Kubernetes cluster on page 1455.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Verified the version of the supported assets and resources that you want to use. See Supported versions of assets and resources on page 705.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.
- Ran API Suites with tests that run on the virtual service resources that are configured.

About this task

After you run the virtual services that are in a project, you can view details of the usage statistics of the virtual services from the **Usage** page on Test.

Restriction: The usage statistics of virtual services if they are run on Test Integrations and APIs agents, are collected only if the agents are of the same version as Test.

Important: The settings such as the **Period**, or **From** and **To** that you configure to view the usage details are retained for you during your ongoing session and the settings are retained even when you perform the following activities:

- Log out of the server and log in again.
- Log in after a server restart.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Usage** in the navigation pane.

   The **Virtual Service Usage** page is displayed.

   The total requests that were received by all virtual services on the server are displayed as a graph.

   The default view displays the usage data of the virtual services that are gathered during the previous week and grouped in an hourly interval. The total requests received from when the virtual service was started are displayed.

   The metric data that is reported is for the number of requests that are received by a virtual service. The requests can be requests, messages, or other calls that are received by the virtual service.
4. Change the default settings to view the graph for the period for which you want to view the usage.

   You can change the default view by selecting an option from the following options:

Option	Description
During the last Hour	Displays the usage data of the virtual service that are gathered during the previous hour and grouped in a 20-second interval.
During the last 24 hours	Displays the usage data of the virtual service that are gathered during the previous day and grouped in an eight-minute interval.
During the last 7 days	Displays the usage data of the virtual service that are gathered during the previous week and grouped in an hourly interval. **Note:** This is the default view.
Custom Range	Displays the usage data of the virtual service that are gathered during the date range specified in the **From** and **To** fields. You can select the date and time for both the **From** and **To** fields. The data is dynamically grouped in intervals that depend on the range selected.

   Note: You can refresh the view at any time to present the graph for the latest data by clicking **Refresh**.
5. View details of the virtual service by hovering the mouse over any of the data-points to view the details at that point.

The following details about the virtual service are displayed when you hover the mouse over the data-point:

- The date and time of the request received.
- The number of requests received.

Results

You have viewed the usage details of the virtual services in your project that ran on Test.

What to do next

You can perform any of the following tasks:

- You can view the routing rules defined in the virtual service, agent, proxy, or intercept. See Viewing routing rules of the virtual services on page 1494.
- You can stop the virtual service instances that are running. See Stopping virtual services on page 1498.

Related information

Management of virtualized services on page 1453

Stopping virtual services

After you started instances of the virtual services from the **Resources** page on Test , you must stop the running instance from the **Instances** page when you do not want to use the instance. You can stop a single instance, multiple, or all running instances.

Before you begin

You must have started the virtual service from the **Resources** page. See Configuring a run of a virtual service on page 1462.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Open the project that contains the virtual service resources in the test assets by clicking **My Projects > *project_name***.

   The **Overview** page is displayed.
3. Go to the **Instances** page and identify the instance of the virtual service that you want to stop.

   You can select the virtual service instances that satisfy the following conditions:
   - The virtual service instance must be either in the `In transition` or `Running` state.
   - The **Stop** icon in the **Actions** column must be enabled for the virtual service instance.
   - The checkbox icon must be enabled for selection for the virtual service instance.
4. Perform the actions indicated in the following table:

If...	Action...
You want to stop any virtual service instance that is running	Perform any of the following actions: ◦ Click the **Stop** icon in the **Actions** column of the virtual service instance. The **Stop virtual service instance** dialog is displayed. ◦ Select the checkbox in the row of the virtual service instance, and then click **Stop selected**. The **Stop all selected virtual service instances** dialog is displayed.
You want to stop multiple virtual service instances that are running	Perform any of the following actions: ◦ Select the checkbox in the row of the virtual service instance, and then click **Stop selected**. ◦ Select the **Select all** checkbox in the header row, and then click **Stop selected**. **Note:** All running instances are selected when you select the **Select all** checkbox. The **Stop all selected virtual service instances** dialog is displayed.

5. Click **Ok**.

   A notification is displayed that the running virtual service instances are stopped.

Results

You have stopped a single instance, multiple, or all running virtual service instances. The stopped virtual service instance on the **Instances** page is displayed with the status as *Stopped by User* in the **State** column.

What to do next

You can restart the virtual service by completing the configurations that you want from the **Resources** page.

You can open and view the `Main container log` of the stopped virtual service.

Test results

After the tests, Suites, or schedules are run and completed, you can view the results and open the reports in Test to analyze the verdict, the performance, and statistics. You can also re-execute tests and schedules from the **Results** page with the same commit id.

Summary

The **Results** page displays the verdict summary for all the tests run in the project.

In this page, you can perform any of the following tasks:

- Displaying the suites only, or the tests only, or tests and suites that are run with advanced settings.
- Searching for test results. You can filter test results by using any of the following ways:
  - By using the **Search** field to search for results by name. When you search by name, the number of rows that are matching with the search keyword are displayed next to the **Search** field.
  - By selecting predefined time interval from the **Selection interval** list.
  - By selecting **To** and **From date** from the **Date Interval** option.
  - By clicking verdict from the **Verdict summary** slider (*Pass*, *Fail*, *Inconclusive*, or *Error*).
  - By creating filter queries.
- Configuring the information displayed on the **Results** page from the **View Settings** window by clicking the **Settings** icon .
- Downloading the result list as a `.csv` file by clicking the **Download** icon .
- Adding labels if you have a tester or an owner role.
- Adding text in the **Notes** field. You can also group the notes similar to a conversation in a chat channel.
- Locking test results: Locked results can be unlocked by the project owner or the project member who locked the results.
- Deleting test or suite results.

  Note: When you delete the result of a suite run, the results associated with the tests and suites within that suite are also deleted. Test results that are part of a suite run cannot be selected individually. To delete such results, you must select and delete the parent test suite run.

- Comparing performance reports.
- Viewing trending reports.
- Viewing multiple reports depending on the test types and the parameters set when you execute the tests: Statistics reports, Functional Test reports, Test logs or Result data queries
- Re-executing tests with the same commit id.

  For more details, see the links in the next section of this page.

If you added any additional properties under the **Additional Configuration Parameters** section in the **Execute test asset** window, then these properties are displayed in the following sections after the test run:

- The **Properties** column on the **Results** page.
- The **Summary** section on the **Functional Report** page.
- The **Properties** section on the **Test log** page.

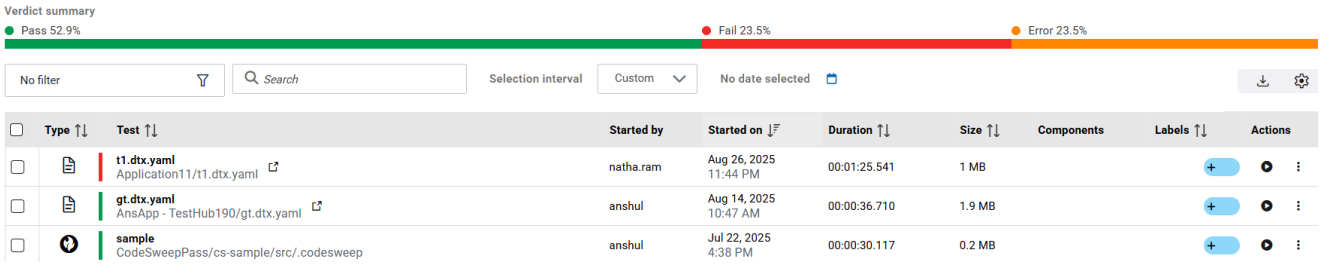
Test results and reports

Test provides a single location for hosting the results and reports of all tests run on different desktop clients and for tests run from the server.

Test results list

When a test run is complete, you can see the results on the **Results** page that you open from **Analyze > Results** in the navigation pane.

The list of test results is organized based on the test type. You can display the suites only in the **Suites** tab, or the tests only in the **Tests** tab, or the tests and suites that are run with advanced settings in the **Advanced** tab.



Results, reports, and logs are generated for tests run from Test, or from desktop clients such as Test Performance, Test UI, or Test Integrations and APIs.

You must have configured the desktop clients to publish reports of tests that are run from the desktop client to the Test. For more information about the publishing procedure, refer to the links at the end of this page.

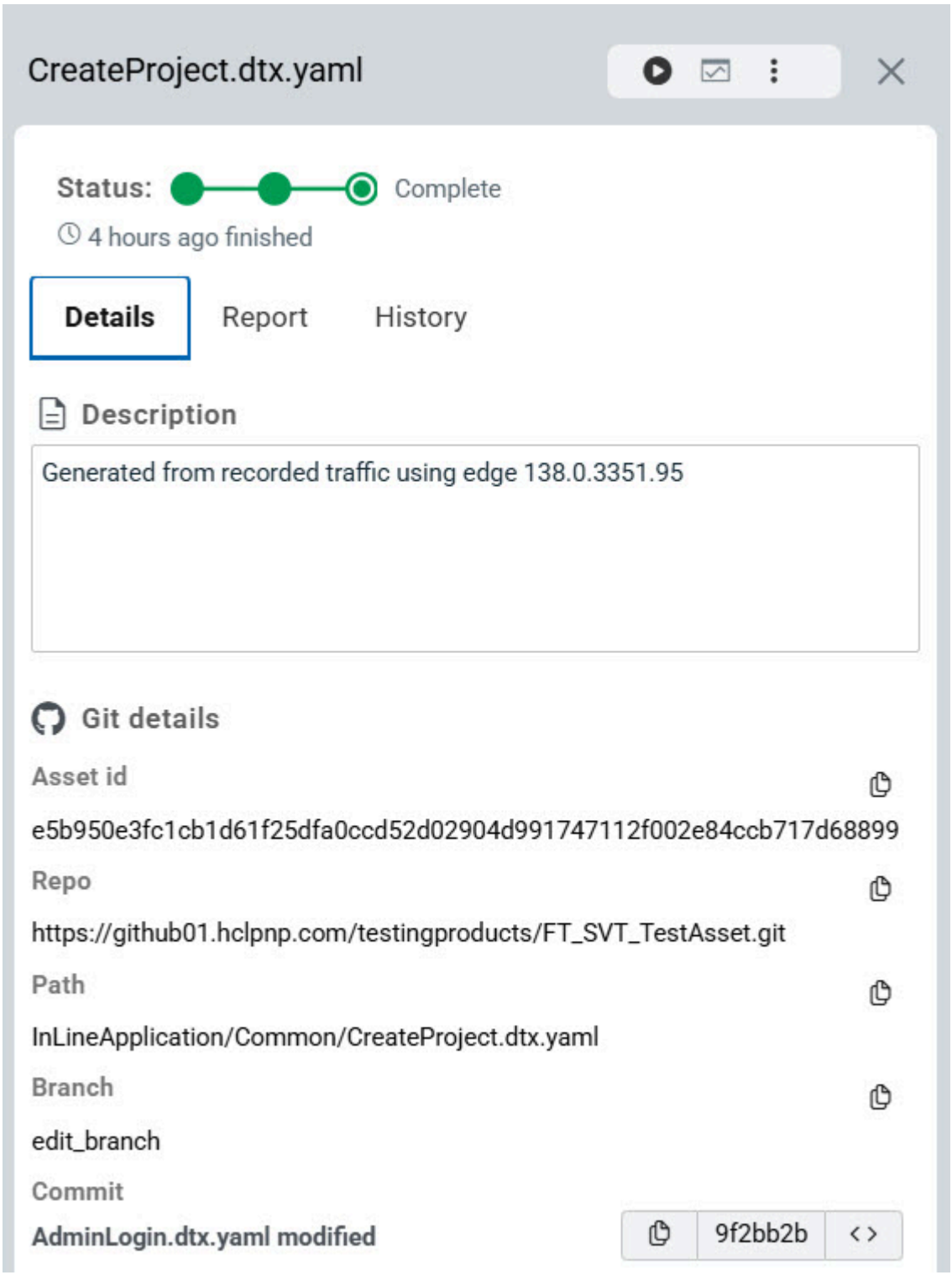
When you click the test result, a results panel is displayed with information in the following tabs:

- Details
- Reports
- History

Test result details

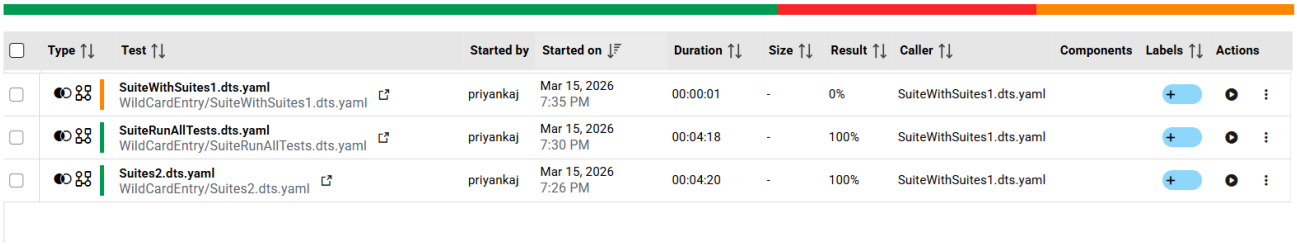
You can view the following details about the test assets used in the test run in the **Details** tab:

- The status of the test results, the date and time it was executed and completed.
- The details about the Git repository that contains the test assets.



You can configure the information displayed on the **Results** page from the **View Settings** window. You access this window by clicking the **Settings** icon .

Depending on the attribute you selected in the **View Settings** window, the selected attributes are displayed in the columns of the **Results** page.



The **Result** page supports selecting individual items from the list. When a row is selected, the corresponding high-level information is displayed in the top banner of the Test Editor.

The **Result** column displays the pass percentage for a test or suite. For a test, when you hover over the percentage, you can see a description indicating the number of steps that passed. For a test suite, the description indicates the number of tests that passed.

The **Caller** column displays the name of the suite that invoked the result.

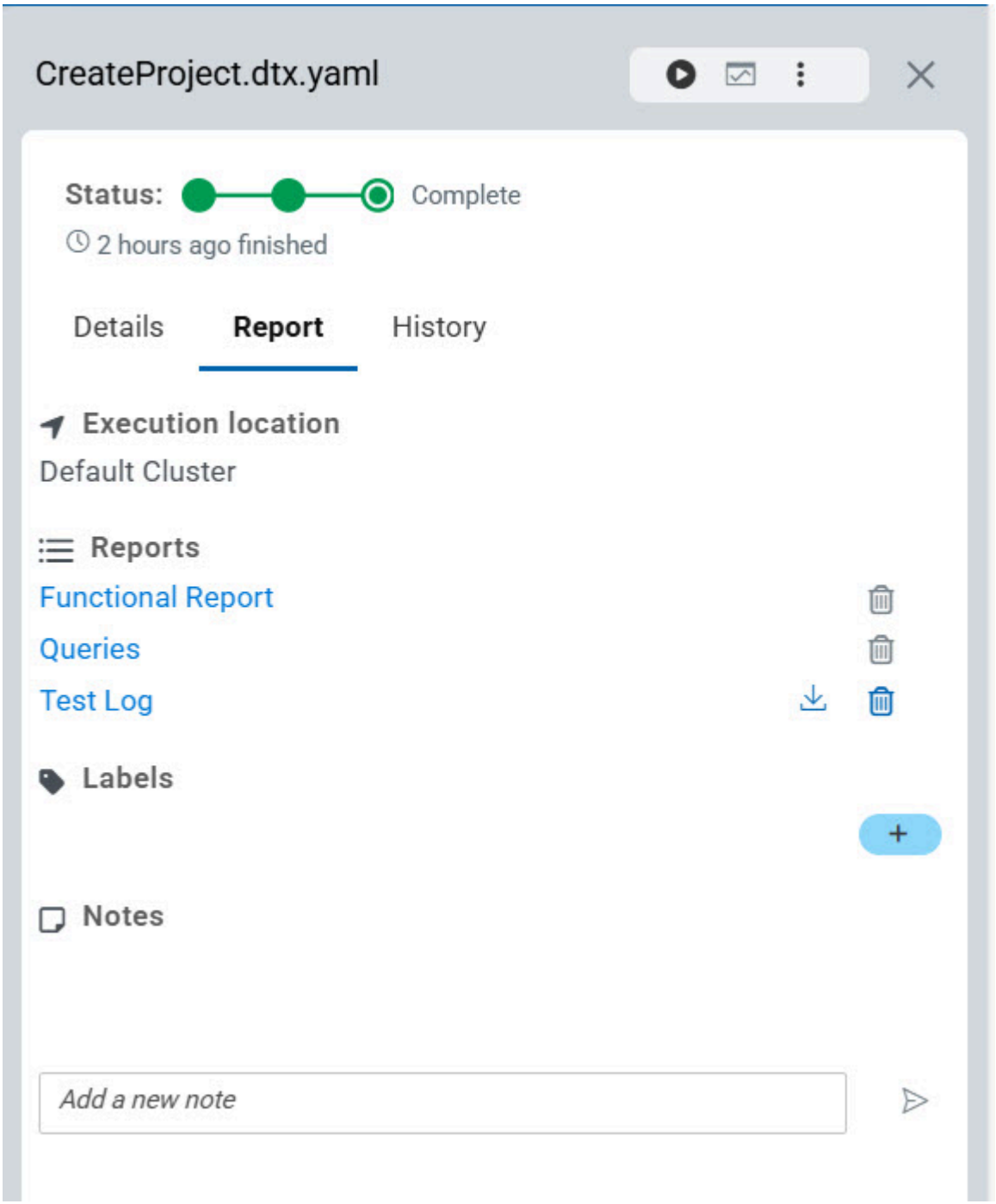
The icon in the **Type** column indicates that the test or suite was run as part of another suite. Hover over the icon to view the name of the suite in which the test or suite was run.

You can click the **Download** icon to download the list displayed on the **Results** page. The result list is downloaded into your local file system as a Comma Separated Values (CSV) file.

Reports

The **Report** tab provides the following information:

- The location where the test was run.
- The reports that are generated for the test run.
- The labels that you created for the test during its configuration for a run.
- Notes that you can add to the specific test run result or displays the notes already added by you or others.
- The size column displays the total disk usage of each test result. The size is rounded up and displayed in human-readable units. Hover over the size to view the exact disk usage. You can also sort by size.



You can perform the following actions on the reports and logs that are displayed on the **Reports** tab:

- View the report that opens in a web browser by clicking the report.
- Download the report by clicking the **Download** icon . You can enter a name and select the location on your computer where you want to save the report.

  The report is downloaded in any of the following formats:

- A compressed file and contains the HTML pages for the supported tests.
- A compressed file that contains the Surefire report.
- A PDF for the supported tests.

Note: For tests that contain more than 30 test steps, the details of the test steps are not included in the PDF.

- Delete the report by clicking the **Delete** icon .

Note: The deleted report is not available on the server. You might have to run the test again to generate the report.

As a default configuration, test logs are delivered in a traditional format for the executed Compound Tests and schedules.

The reports are generated for Compound Tests, Schedules, and Test Suites, or for other tests that are run from a desktop client or from Test.

The following table lists the reports that are generated for each type of test and the links to help pages.

Table 616. Analyzing Reports

Report	Tests type	Product	More information
Functional Report	Server Test	Test	Viewing a Functional Report on page 1511
	Server Suite		
	API Test	Test Integrations and APIs	
	API Suite		
	Compound Test		
	Functional test	Test UI	
	AFT Suite		
	Compound Test		
Functional Test Report	API Suite	Test Integrations and APIs	Viewing reports
	Compound Test	Test UI	Unified Report
	AFT Suite		
	Functional test (For Web UI tests)		
Mobile and Web UI Report	Compound Test	Test UI and Test Performance	Mobile and web UI test result reports

Table 616. Analyzing Reports (continued)

Report	Tests type	Product	More information
	AFT Suite	Test UI	
Statistics Report	Compound Test	Test UI and Test Performance	Reports and counters. Service Performance report. From the Statistic reports, you can add counters. Refer to Adding additional counters on a separate page.. You can also display the statistics of counters on graphs. Refer to Displaying counter data in tables or as graphs..
	AFT Suite	Test UI	
	VU Schedule	Test Performance	
	Rate Schedule		
	Functional test (For Web UI tests)		
Test Log	Compound Test	Test UI and Test Performance	
	AFT Suite	Test UI	
	API Suite	Test Integrations and APIs	
	Functional Test (Web UI)	Test UI	
	VU Schedule	Test Performance	
	Rate Schedule		

Publishing test results to DevOps Test Hub

Publishing specific results to the server

Publishing test results to the server

Publishing functional reports to the server

Publishing specific results to the server

Publishing test results to DevOps Test Hub

Publishing existing test suite results to DevOps Test Hub

Creating search queries to filter test results

To filter the test and suite results list, you can create a search query that you apply as a filter in Test .

About this task

To filter the list of test and suite results, you can use rules and group of rules in your query. This procedure explains how to create a query that contains two rules and a group of rules as an example, and how to select other filters.

1. Click **New filter** on the Result page.

Follow these steps to add a rule in your query:

2. Select **Add a rule** in your query, and enter or select the required attribute in the fields to define the rule:

    a. Select the first attribute that you want to query in the drop-down list.
    Example
    Select **Verdict**.

    b. Select the condition as *equal to (=)* or *not equal to (!=)*, *Contain* or *Not contain*.
    Example
    Select **=**.

    c. Select a value for the first attribute you selected.
    The content of this field depends on the option you selected as first attribute. Enter a value for the **id**, **TypeTest**, **Version**, **Result**, **Labels**, **Branch**, **Test path**, and **Text** attributes. For the other attributes, select a value in the drop-down list.
    Example
    Select **Fail**.

    d. Select either the **AND** or the **OR** operator in the first field to add a condition to compare the first rule with another rule.

    e. Select **Add rule** in the drop-down list.

    f. Select an attribute, a condition, and select or enter a value.
    Example
    Select **Test** as an attribute, select **=** as a condition operator, and enter *mytest* as a value.

Follow these steps to add a group in your query:

3. Select **Add group** in the drop-down list.

   Note: A group is a collection of rules and can also contain other groups within a query. You can add a rule or a group within this group.

4. Select a condition operator.

   Example

   Select **Or**.

5. Select **Add rule** in the drop-down list to add a rule in the group.
6. Select the required an attribute, an operator, and enter a value for the rule.

   Example

   Select the **Type** attribute, **=** as an operator, and **Compound Test** as a value.

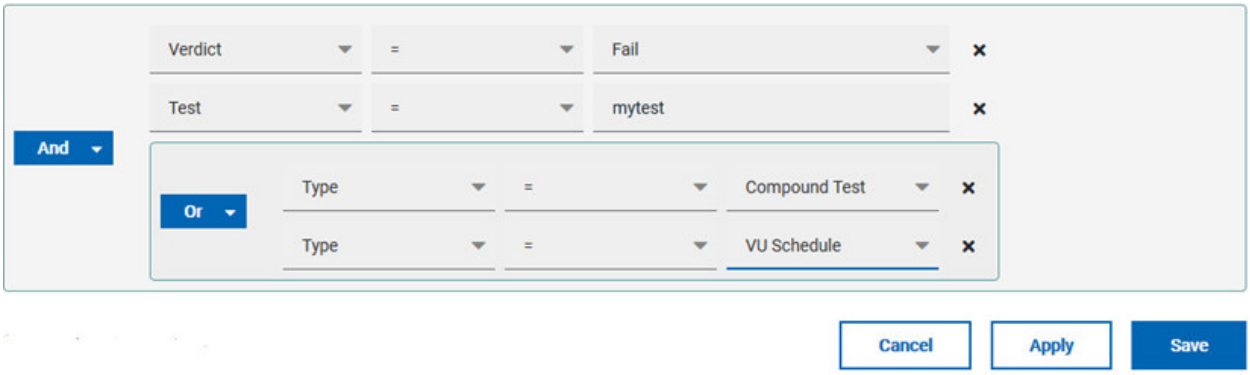
7. Select **Add rule** to add another rule in the group.
8. Select and enter the appropriate parameters as you did for the other rules.

   Example

   Select **Type** as an attribute, **=** as a condition, and **VU Schedule** as a value.

9. Click **Apply** if you want to apply the filter now.

   You can cancel the query or save the query when the results are found.



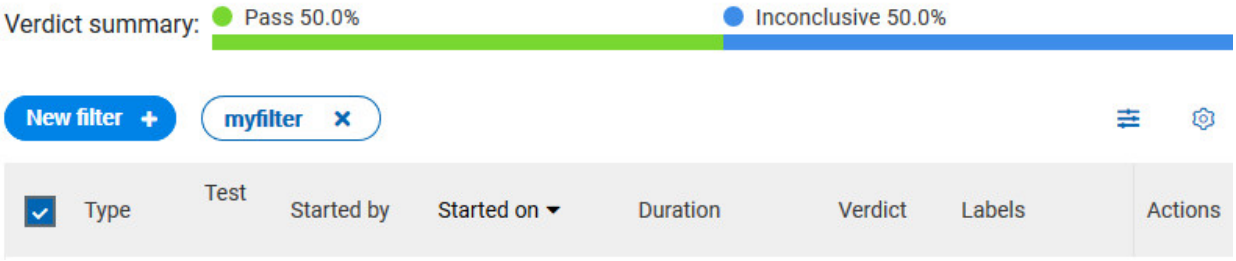
10. Click **Save**.

Proceed as follows to select your filters:

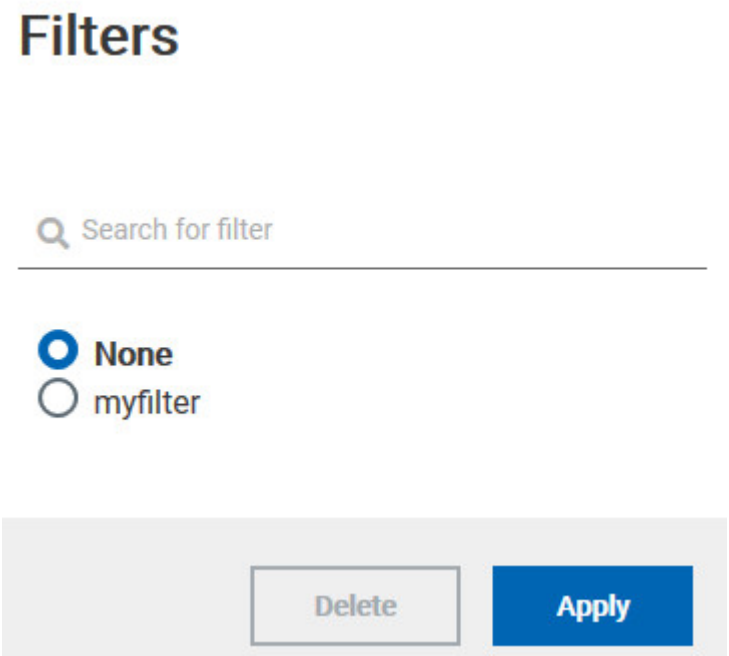
11. Enter a name for the filter and save in the dialog that is displayed.

    Result

    The filter is displayed as a favorite filter.



12. Click the icon to select the filters you created from the list of saved filters.



13. Click a filter in the list of saved filters. If a filter does not display in the list, enter the name of a filter in the search field and select it.

Note: You can delete a filter from the list or disable filters by selecting **none** in the list.

14. Apply the selected filter.

Result

The filter is displayed as a new favorite filter in the **Results** page.

Comparing test reports

After you run performance tests, you can analyze the difference between two or multiple test results in Test . For example, you can compare the performance of an application at different time slots or different milestone builds between two test results.

About this task

You can compare the test runs for performance tests that are in the same project or in different projects.

1. Select results in the **Results** page by checking the boxes in the first column.
2. Click the icon

Note: When comparing multiple runs, you cannot compare multiple time-ranges or stages.

Result

The two or multiple test results are displayed in a browser window in the same report.

Viewing trending reports

When the performance test runs are complete, you can view the trend of response time for an application over a period of time in Test from trending reports. In addition to the response time, you can view the trend for the loops, transactions, and performance requirements for the application.

About this task

The trending reports are available for performance tests only.

1. Select results in the first column in the **Results** page.
2. Click the icon to view the trending reports.

Note: When comparing multiple runs, you cannot compare multiple time-ranges or stages.

Results

You can analyze the trend information for multiple test results in the same report from a browser window.

Viewing a Functional Report

When the test run is complete, you can view the **Functional Report** in Test . The report contains a summary of the execution, the test environment, a detailed view of all tests, and the test steps that you executed.

1. Go to **Analyze > Results** in the navigation pane.

   The test assets that have completed their run are listed.

2. Select the tab based on the type of test assets that you ran as indicated in the following table:

If...	Then...
You ran AFT Suites, API Suites, Server Suites, or Compound Tests.	Click the **SUITES** tab.
You ran API Tests, Functional Tests, or Server Tests.	Click the **TESTS** tab.
You ran Suites or Tests and you want to find the test assets by using custom filters.	Click the **ADVANCED** tab.

3. Click the test asset under the **Test** column.

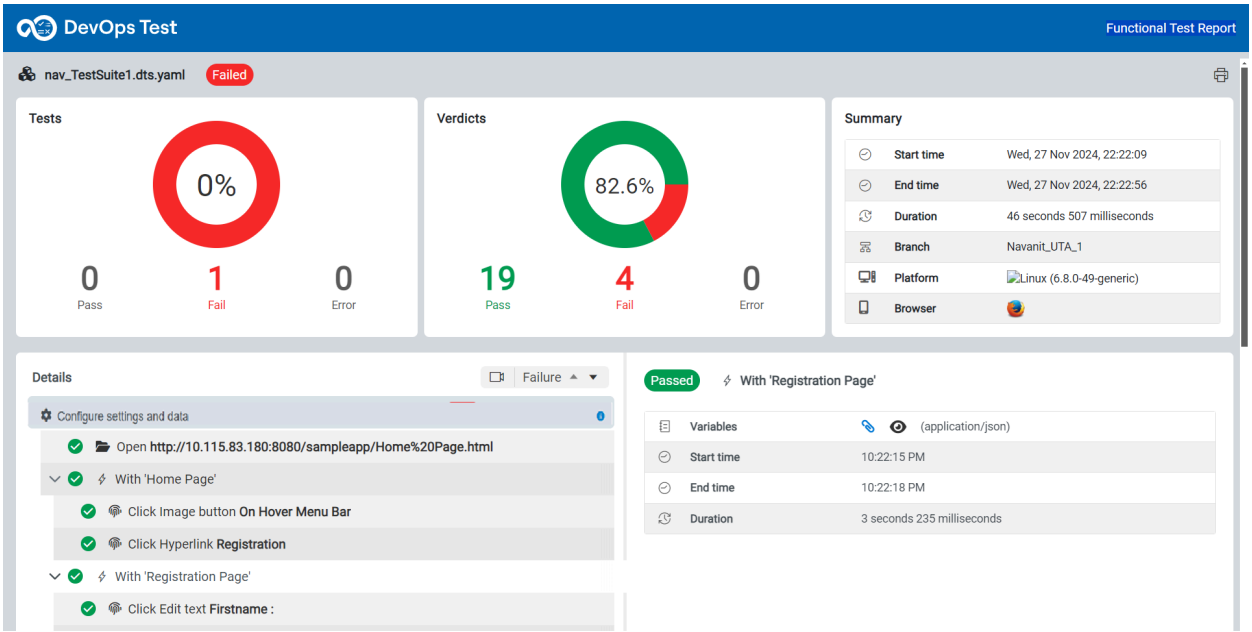
   The results and reports panel is displayed.

4. Click the **Reports** tab, if it is not already open.
5. Click the **Functional Report** link.

   The **Functional Report** opens in a new tab.

   You can see the following information in the report:

   - Click the **Configure settings and data** node to view the initial variable values and dataset details used during test run. Variable details are now displayed directly in the parent activity viewer for better readability and navigation.
   - The top-level steps and first-level child steps are automatically expanded to provide better visibility.
   - A summary of the test run, including passed or failed status, count of passed or failed tests or test steps, and the verification points.
   - A summary of the test environment with the details of deployments, datasets, and components.
   - A detailed view of all tests that were executed in the case of an intelligent test suite, or in the case of a test with invoked tests.
   - A detailed view of all steps that were executed for each of the tests, including screenshots if any.



6. In the report, you can perform the following actions:

   a. Click the **Print** icon to print the report or save it as a PDF.

   b. Click the **Browse Screenshots** icon to view all the screen captures taken during the test run.

   c. Click the **View content** icon to view the variables or stack trace, if available.

d. Click the **Download attachment** icon to download the variables or stack trace as a `.txt` file, if available.

e. Click the **Previous failure** icon and **Next failure** icon to navigate between the failed steps, if there are failed steps in the report.

Results

You have viewed the **Functional Report** and analyzed the summary of the tests that you ran.

Navigating to a test, suite, or step from the functional report

You can navigate to the related test, suite, or specific step from the functional report to investigate failures, review details, or make corrections. By using this feature, you can locate test resources faster and fix issues quickly.

Before you begin

You must have run a test or suite in Test.

1. Go to **Analyze > Results** in the navigation pane.

   The test assets that have completed their run are listed.

2. Select the tab based on the type of test assets that you ran as indicated in the following table:

If...	Then...
You ran AFT Suites, API Suites, Server Suites, or Compound Tests.	Click the **SUITES** tab.
You ran API Tests, Functional Tests, or Server Tests.	Click the **TESTS** tab.
You ran suites or tests, and you want to find the test assets by using custom filters.	Click the **ADVANCED** tab.

3. Click the test asset under the **Test** column.

   The result and report panel is displayed.

4. Click the **Reports** tab, if it is not already open.
5. Click the **Functional Report** option.

   The functional report page opens in a new tab.

6. Click the expand icon to view the result details in a functional report., then

   The individual test steps or suite results are displayed

7. Click the **Open** icon to access the specific test, suite, or a step in the **Test Editor**.
   If the selected test, suite, or a step is on a different branch, the **Open Asset** dialog prompts you to switch to the appropriate branch before opening the asset in the **Test Editor** page.

   If the selected test, suite, or a step is on the same branch, it opens directly in the **Test Editor** in a new tab.

   Note: The **Open** icon is disabled if the test, suite or a step is deleted or renamed.

Results

You have directly navigated to a selected test, suite, or specific step from the functional report.

Re-executing tests from results

You can re-execute a test, a schedule, a collection of tests or schedules from Test if you want to retain the same commit id.

Before you begin

You must have executed a test or multiple tests from the **Execution** page in Test.

Re-executing a test from the **Results** view requires a Tester or an Owner role.

About this task

The following procedure describes how to re-execute a single test or multiple tests at a time.

Perform all the following steps from the Results page.

- Proceed as follows to re-execute one test at a time:

  1. Identify the test that you want to re-execute and click the **Re-execute** button in the Actions column.
     Result
     The **Execute test asset** dialog displays the same parameters as the ones that were set to execute the initial test from the **Execution** page.

  2. Select another **Version** of the test.

     Note: If you select another version of the test, the settings configured in the dialog and in the **Environment**, **Data Sources** and **Locations** tabs might change.

  3. Modify the **Schedule** parameters.

Note: If you don't want to execute the test now, schedule the time when you want the test to be re-executed.

4. Click the **Environment** tab and select another **API test environment** if multiple environments were initially set for a test that is running an API Suite.
5. Click the **Data Sources** tab and select an override option if multiple data sources were defined for the initial test run.
6. Click the **Variables** tab and add a new variable if you want to re-execute a test asset with a different variable from the one configured in the asset.
7. Click the **Location** tab to override the default cluster location if the test asset has static agents configured.
8. Click **Advanced** to modify the advanced parameters that were set for the initial test run.
9. Enter a new instance of **Java arguments** if applicable to the test.
10. Enter the environment variables that must be passed to the test run at runtime in the **Environment Variables** field.
11. In the **Resource Monitoring** tab, click the , and press the Ctrl + Space keys, or enter the initial letter of a label to select a label in the list and override the current sources in the schedule.

    Note: The **Resource Monitoring** tab is displayed for performance schedules and only if you have enabled the **Resource Monitoring from Service** option from Test Performance.

    The label field is grayed if you don't have any label entered in your current project. To collect data during the test execution, you must first enter labels in the Resource Monitoring sources page from Test.

    For more details about controlling Resource Monitoring sources and labels, see the links at the end of this page.

12. Click the **Re-execute** icon to re-execute the test now or at a scheduled time.

- Proceed as follows to re-execute multiple tests at a time:

1. Click the checkboxes for the tests that you want to re-execute.
   You can select them one by one or select all of them.
   Result
   A toolbar is displayed.
2. Click the **Re-execute** icon in the toolbar.

Result

A message indicates the number of selected tests and the number of tests to be re-executed. The tests are re-executed with the same initial commit ids and parameters.

3. Click **Execute**.

Result

You can view the results, logs, and reports of the re-executed tests in the **Results** page.

Related information

Creating a project on page 1728

Test results and reports on page 1501

Controlling resource monitoring sources in a schedule on page 1595

Test run configurations on page 1134

Test results data in Grafana

By using Test Results Data and Grafana, you can retrieve and explore test log data that is generated in Test. You must install and configure the Test Results Data plugin in Grafana and create an instance of a data source that you use to explore your test log data.

Grafana is a visualization and analytics software. You can use Grafana to explore Test metrics.

You must use Test Results Data, which is a data source plugin, to make requests to Test and represent data in a Grafana dashboard.

Installing and configuring the DevOps Test Hub Results Data plugin

You must install and configure the DevOps Test Hub Results Data plugin that is packaged along with Test to view and analyze the log data in Grafana. You can view the test log data that is generated when you run a test from Test in a Grafana dashboard.

Before you begin

You must have installed Grafana on your computer.

1. Perform the following steps to install the DevOps Test Hub Results Data plugin:
    a. Download the DevOps Test Hub Results Data plugin from the following URL:

    ```
    <your server address>/test/artifacts/grafana/devops-testhub-datasource.zip
    ```

    b. Install the DevOps Test Hub Results Data plugin by extracting the compressed file into your Grafana plugin directory.
    c. Restart the Grafana server.
2. Perform the following steps to configure the data source by using the DevOps Test Hub Results Data plugin:

a. Enter the address of the local Grafana server in your web browser:

Important: You must sign in with `http://localhost:3000` or `http://localhost:8080` URLs to use the plugin.

b. Follow these steps to verify that the DevOps Test Hub Results Data plugin is available in Grafana:
   i. Click **Administration > Plugins and data > Plugins** in the left navigation panel of Grafana.
   ii. Enter **DevOps Test Hub Results Data** in the search bar in the **Plugins** tab.
   iii. Click **DevOps Test Hub Results Data** in the search results.

   The DevOps Test Hub Results Data page is displayed. You can find the links to Test documentation and the description of the plugin on this page.

c. Perform the actions indicated in the following table to create an instance of a data source in Grafana by using the DevOps Test Hub Results Data plugin on the **DevOps Test Hub Results Data** page:

Note: Users with the role of a server administrator can add data sources.

Options	Actions
If you want to create an instance of the data source in Grafana by using the proxy connection	Perform the following steps: i. Click **Add new data source**. The data source configuration panel is displayed. ii. Click **Default** if you want to set the data source as the default data source in Grafana. iii. Enter a name to identify the data source. iv. Enter the URL of Test in the **Server** field. v. Enter the token that you created in Test. **Note:** The **Use proxy** is enabled by default. vi. Click **Save and test**. **Note:** If the server certificate is self-signed, an error message is displayed. You must add the CA certificate to establish the connection with the server. vii. Perform the following steps to add the CA certificate: 1. Enable **With CA Cert** by using the toggle icon. 2. Add the CA certificate in the **CA Cert** field that you saved during the installation of Test. 3. Click **Save and test**.

Options	Actions
If you want to create an instance of the data source in Grafana with additional Helm parameters **Important:** Before you install Test, you must use an additional Helm command parameter **router.allowedOrigin** during the installation. You can also run this command after installation. See the Additional Helm parameters topic in the related information section.	Perform the following steps: i. Click **Add new data source**. The data source configuration panel is displayed. ii. Click **Default** if you want to set the data source as the default data source in Grafana. iii. Enter a name to identify the data source. iv. Enter the URL of Test in the **Server** field. v. Enter the token that you created in Test. vi. Disable **Use proxy** by using the toggle icon. vii. Click **Save and test**.

Results

You have installed and configured the DevOps Test Hub Results Data plugin in Grafana.

What to do next

You can retrieve the test log data from Test by using queries.

Related information

Additional Helm parameters

Download Grafana

Install Grafana plugins

Access management for DevOps Test on page 1709

Add a data source

Explore

Viewing test results data in Grafana on page 1519

Viewing test results data in Grafana

After you install and configure DevOps Test Hub Results Data plugin in Grafana, you can retrieve the test log data that is generated in Test and view the test results data in Grafana. You can use queries to display graphs, statistics, or tables. You can also view the details in a Grafana dashboard.

Before you begin

You must have completed the following tasks:

- Ran tests to get test logs results in Test. For more information about the procedure, refer to Test logs on page 1523.
- Installed the DevOps Test Hub Results Data plugin and created an instance of data source for DevOps Test Hub Results Data. See Installing and configuring the DevOps Test Hub Results Data plugin on page 1516.
- Opened Grafana with one of these ports: `http://localhost:3000` or `http://localhost:8080`.

1. Click **Explore** in the **Toggle** menu in Grafana.

   The **Data query** page with the default data source is displayed.
2. Select **DevOps Test Hub Results Data** from the list of data sources.
3. Select the type of query by performing any of the following actions when you want to view statistics, events, or activities from the test data:

To view	Actions
The variation of a metric over time.	Perform the following steps: a. Click the time-series statistics tab icon . b. Go to Step 4 on page 1519.
The values of the metric computed over the entire time range.	Perform the following steps: a. Click the overall statistics tab icon . b. Go to Step 4 on page 1519.
The events recorded in the test log.	Perform the following steps: a. Click the **Events** tab. b. Go to Step 4 on page 1519.
The activities recorded for all events in the test log.	Perform the following steps: a. Click the **Activities** tab. b. Go to Step 4 on page 1519.

4. Define the scope for the query by performing the actions indicated in the following table:

Options	Actions
If you want to use the data from the team space	**Note:** You can select the team space as the scope for your query only if you want to view the time-series statistics or overall statistics. Perform the following steps: a. Select the **Team Space** option. b. Select the team space from the list of team spaces that are listed. c. Select either **Service**, if you want to use Resource Monitoring data or **Stubs**, if you want to use the virtual service data in the query. d. Go to Step 5 on page 1520.
If you want to use the data from your project	**Note:** You can select the project as the scope for your query if you want to view any of the supported query types. Perform the following steps: a. Select the **Project** option. b. Select the project from the list of projects that are listed. c. Select the **Result** option. d. Select the result from the list of results that are displayed. Alternatively, you can also enter text for the name or ID of the result in the **Search** field, and then select the result from the listed items. e. Go to Step 5 on page 1520, if you selected the query type as time-series statistics or overall statistics, or go to Step 6 on page 1521, if you selected the query type as **Events** or **Activities**.

5. Perform the following steps to select metrics for your query:

   a. Select a metric from the list of metrics in the row of the **Metrics** option.

   b. Assign a name for the metric that helps you identify it, by entering text in the **Metric** field.

   c. Select the aggregation factor for the metric. For example, select **Sum** to add all values of the metric.

   d. Optionally, click the icon to add a separator.

      With separators, you can view details of a specific project or test result in comparison with the values of the metric across projects or results in the team space or project.

   e. Optionally, click the **Cumulated** toggle-button if the values for the metric must be added cumulatively over time. The default option is set to be non-cumulative.

f. Add another metric by clicking the **Add metric** icon .

Note: If you added multiple metrics and you want to remove a metric, you can click the **Remove metric** icon of the metric.

g. Go to Step 7 on page 1521.

6. Perform the following steps for the **Events** or **Activities** query type:

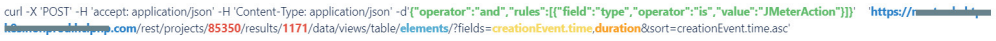
a. Enter the **Query Filters**.

b. Enter the **Timing value** and **Value**.

The timing values are sorted in ascending order. The value must be numeric so that it can be used in a graph.

Tip: To find all the information needed on the **Explore** page of DevOps Test Hub Results Data, perform any of the following actions:

- Select a result on the **Queries** page in Test and copy the result number, query filters for the events or the activities, query filters, and timing values. You can find the project number in the URL of the **Queries** page.
- Click **Copy curl command** on the **Queries** page in Test and paste the curl command in a text editor. You can find all the information you need in the following command.



7. Click **Add query** if you want to combine queries from different projects, results ids, or with different properties in the query filters.
8. Click **Run query**.

Results

You configured a query to get the test log data from Test, and you can view and analyze the log data displayed for your query.

What to do next

You can build a dashboard including graphs, charts, gauges, tables to get customized visual representations of your test result data and select options to display the test result values. See Dashboard overview for more details about dashboards in Grafana.

Viewing Surefire format reports

After you execute JMeter, JUnit, API, and Functional tests with advanced settings in Test , you can view the test results in a Surefire format report by using dedicated URLs.

Before you begin

You must have configured your tests with one of the following parameters:

- For JMeter, JUnit, Postman, and HCL® AppScan Suites, set the **history** parameter with **cisterna** value in the **Additional Configuration Parameters** field in the **Advanced** settings.
- For API Suites, AFT Suites, Compound Tests, Functional tests, and API tests, set the **-history cisterna** parameter in the **Program Arguments** field in the **Advanced** settings.

About this task

When a test run is complete, you can see the test result data in a report that is in a Surefire format. You can view this report by using dedicated URLs.

1. Click **Analyze > Results** in the navigation pane.
   Result
   The **Results** page is displayed.
2. Click a test result, and note its project identifier and test result identifier (ID) in the URL of the test log that is displayed in your browser. The URL is in the following format: `https://myProductionServer.com/test/#/Initial/project/<ProjectID>/analyze/test-log/<ResultID>`.
   The identifiers are required to view the Surefire test logs in the following steps.
3. Enter one of the following URLs with the appropriate information in a web browser or in an external tool that you use to view Surefire reports:
   - `/rest/projects/<ProjectID>/data/views/surefire/`

     Enter your REST API address and the project identifier to view the list of all test cases in the project with details about their status.

     Example: `https://myProductionServer.com/rest/projects/230200/views/surefire/`
   - `/rest/projects/<ProjectID>/results/<ResultID>/data/views/surefire/`

     Enter your REST API address, the project identifier, and the test result identifier to view the results for this test result id in the project.

     Example: `https://myProductionServer.com/rest/projects/230200/results/80250/data/views/surefire/`

Results

The test result reports are displayed in a Surefire format.

Related information

Test logs on page 1523

Test log queries on page 1538

Test logs

When you start a run of a test asset, Test starts to log information about the test run in a test log. You can view the test log while the test runs and after the run is completed from the **Test Log** page.

Overview

Test captures the test log for the supported tests that you can run on Test. For the list of supported tests, see Supported suites and tests on page 706.

Test uses a built-in PostgreSQL database to store information about the test while the test runs on the server. Test captures and displays the test log as the default configuration and you do not have to configure other settings during or after installation of the server.

Supported formats of test logs

The formats in which you can view the test logs are as follows:

Table 617. Supported formats for test log display

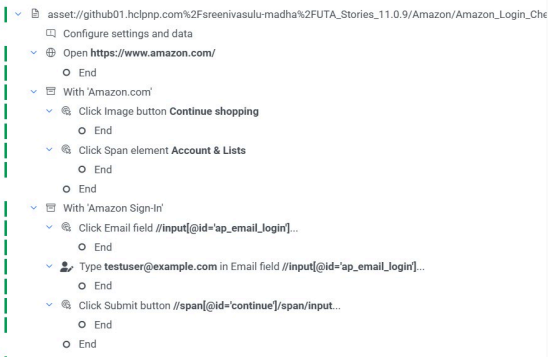
Log format	Description	Action
Default format	Displays the test log in the built-in format on the **Test log** page. For example, the test log can be as follows:  Click the **Configure settings and data** node to view the initial variable values and dataset details used during test run. Variable details are now displayed directly in the parent activity viewer for better readability and navigation. The top-level steps and first-level child steps are automatically expanded to provide better visibility.	You do not have to set any Program argument while you configure the test run.

Table 617. Supported formats for test log display (continued)

Log format	Description	Action

Text format	Displays the test log in a plain text format and is displayed in a different browser tab or window than the one that displays the **Test log** page. For example, the test log can be as follows:
	`----------`
	`Name: autoResults`
	`Description:`
	`Test Version: null`
	`Verdict: fail`
	`Type: com.ibm.rational.test.lt.lttest`
	`----------`
	`----------`
	`Name: DefaultSchedule`
	`Description:`
	`Timestamp: May 25, 2022 at 9:24:10 AM GMT`
	`null`
	`Event type: com.ibm.rational.test.lt.scheduleStart`
	
	`Properties:`
	`Test Log Filter (ALL): 100 (ALL)`
	`Test Log Filter (WARNINGS): 100 (ALL)`
	`Test Log Filter (ERRORS): 100 (ALL)`
	`Type: start`
	`----------`
	`----------`
	`Name: DefaultUserGroup`
	`Description:`
	`Timestamp: May 25, 2022 at 9:24:10 AM GMT`
	`null`
	`Event type: com.ibm.rational.test.lt.usrGroupStart`
	
	`Properties:`
	`Type: start`
	`----------`
	`----------`
		You must set the **-history testlog** Program Argument in the **Advanced** settings dialog while you configure the test run.

Activities and events

The information about the tests that is collated by Test can be classified into the following types:

Table 618. Types of information collated for test logs

Information type	Description
Activity	Specifies the events that are performed when the test runs as part of a test step. An activity comprises different events and is represented as a collection of events that were performed over time. For example, opening a specific web page by using a web browser in the test can be an activity that comprises the different events such as connecting to the web site, sending of a request, and receiving a reply. Each of these events might take a specific amount of time, which are captured in the test log.
Event	Specifies an action performed as part of a test step.

Table 618. Types of information collated for test logs (continued)

Information type	Description
	For example, start connection can be an event in a test that requires a specific web page to be opened in a web browser, as detailed in the example for an activity. The exact time at which this event occurred when the test ran is captured in the test log.

Test log tasks

You can find information about the tasks that you can perform on the the test logs in the **Test log** page. The test logs for the tests that you ran on Test are displayed in the **Test log** page.

The tasks that you can perform on the test logs are summarized in the following table:

Table 619. Test log tasks

Task	More information
Viewing test logs after you start a test run on the **Execution** page.	Viewing test logs after you start a test run on page 1525
Viewing test logs after the test run is completed.	Viewing test logs after the test run is completed on page 1526
Viewing details about the test log.	Test log details on page 1530

Viewing test logs after you start a test run

When you configure and start a test run, you can view the test log that is created progressively while the test runs.

Before you begin

You must have configured and started a test run. See Test run configurations on page 1134.

Note: You must have opened the **Execution** page to start the test run.

1. Click **Show in progress page** icon in the **Actions** column of the test asset on the **Execution** page.

   The asset is displayed on the **Progress** page.
2. Click the **Open action menu** icon.
3. Click the **View test log** option.

The **Test log** page is displayed with the test log collated for test run.

Note: Because the test log is collected progressively as the test runs, you must click **Refresh** to view the latest or the updated test log.

Results

You have viewed the test log for the test asset that you ran.

What to do next

You can view the details of the test log and decide on the actions you might want to take on the test log. See Test log details on page 1530.

Viewing test logs after the test run is completed

After you configure a test run and the run is completed, you can view the test log that is created for the test.

Before you begin

You must have configured and started a test run, and noted that the test run has the status as *Complete* displayed on the **Progress** page.

About this task

You can view the test log that is created for a specific test run from the following pages:

- The **Progress** page. To view the test log from the **Progress** page, go to Step 1 on page 1526.
- The **Results** page. To view the test log from the **Results** page, go to Step 2 on page 1527.
- The **Test log** page. To view the test log from the **Test log** page, go to Step 3 on page 1529.

1. Perform the following steps after the test completes its run:
    a. Go to **Execute > Progress**, if you are not already on the **Progress** page.

        The test assets that are running or have completed their run are listed. If you are already on the **Progress** page, only the test asset that you selected on the **Execution** page is displayed with the status as *Complete*.

    b. Click the **Open action menu** icon ⋮ under the **Actions** column in the row of the test asset.

    c. Click the **View test log** option.

        The **Test log** page is displayed with the test log collated for test run.

        The top-level steps and first-level child steps are automatically expanded to provide better visibility.

Click the **Configure settings and data** node to view the initial variable values and dataset details used during test run. Variable details are now displayed directly in the parent activity viewer for better readability and navigation.

d. Go to Results on page 1530.

2. Perform the following steps after the test completes its run:

a. Go to **Analyze > Results**.

The test assets that have completed their run are listed.

b. Select the tab based on the type of test assets that you ran as indicated in the following table:

If...	Then...
You want to run AFT Suites, API Suites, Compound Tests, Rate Schedules, VU Schedules, JMeter tests, JUnit tests, or Postman tests..	Click the **SUITES** tab.
You ran API tests, Functional tests, or Performance tests.	Click the **TESTS** tab.
You want to find Suites and tests by using custom filters, when you ran Suites and tests.	Click the **ADVANCED** tab.

c. Identify the test asset that you ran by performing any of the following actions:

- Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test asset by entering any text contained in the test asset name in the **Search** field.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab to filter the displayed assets based on the asset type.

  For example, select **API Suite** in the **SUITES** tab to display all API Suites or select **Functional Test** in the **TESTS** tab to display all Functional tests that are in the branch.
- Click the **Filter** icon in the **ADVANCED** tab, and then create a filter query by using the **New filter** option by performing the following steps:

i. Click **New filter**.
ii. Enter a name for the filter.
iii. Select an operator, and then add a rule, or a group of rules.
iv. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

You can select a parameter from the following list:

- Type
- Test Asset Name
- Test Asset Path
- Last Result
- Next Run
- Components

v. Save the filter query to save and apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

- Retrieve and apply a saved filter query, if you have saved filter queries previously by performing the following steps:

Note: The filter query applied previously is selected and the assets based on that filter query are displayed. To apply a different filter query, you must have created and saved the filter query.

i. Click the **Filter** icon in the **ADVANCED** tab.

The filter queries that you created and saved are displayed.

ii. Click the filter that you want to apply.

The test assets that match the filter criteria are displayed.

You have identified the test asset that you ran.

d. Click the test asset under the **Test** column.

The results and reports panel is displayed.

e. Click the **Reports** tab, if it is not already open.

f. Click the **Test Log** link.

The test log created for the test run is displayed in the **Test log** page.

g. Go to Results on page 1530.

3. Perform the following steps after the test completes its run:

a. Go to **Analyze > Test Log**.

   The **Test log** page is displayed. The test assets that were run are displayed.

b. Identify the test asset by its name or the type for which you want to view the test log.

c. Click the test asset under the **Test** column.

   The test log created for the test run is displayed.

d. Go to Results on page 1530.

Results

You have viewed the test log for the test asset that you ran.

What to do next

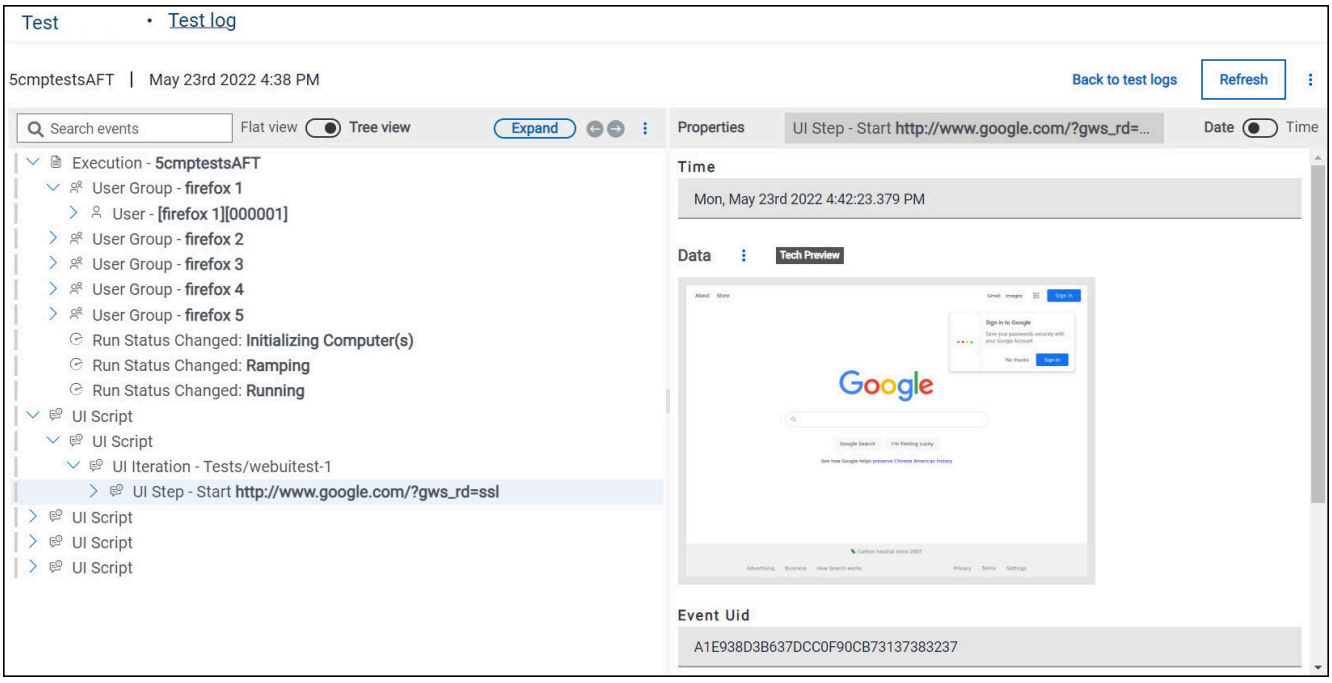
You can view the details of the test log and decide on the actions you might want to take on the test log. See Test log details on page 1530.

Test log details

You can find information about the details that are displayed for a test log on the **Test log** page. You can also find information to use the different options in the UI to change the view of the test log, find details of the events captured, navigate through the log, view or download attachments that are captured.

Before you can view the test log, you must have started a test run and opened the test log that is displayed on the **Test log** page.

An example of a test log that is displayed on the **Test log** page is as follows:



You can find the following details that are displayed on the **Test log** page:

Table 620. Test log details

Details about...	Examples as displayed in the Test log page...
The project name	An example of the project name that is displayed is as follows: Test
The name of the page	An example of the page name that is displayed is as follows: • Test log You can click the Test log link to view the test logs of all tests that ran in your project, if you are viewing the test log of a specific test run.
The name of the test asset	An example of the name of the asset that is displayed is as follows: 5cmptestsAFT
The date and time when the test run was started from the **Execution** page	An example of the date and time when the test run was started that is displayed is as follows:

Table 620. Test log details (continued)

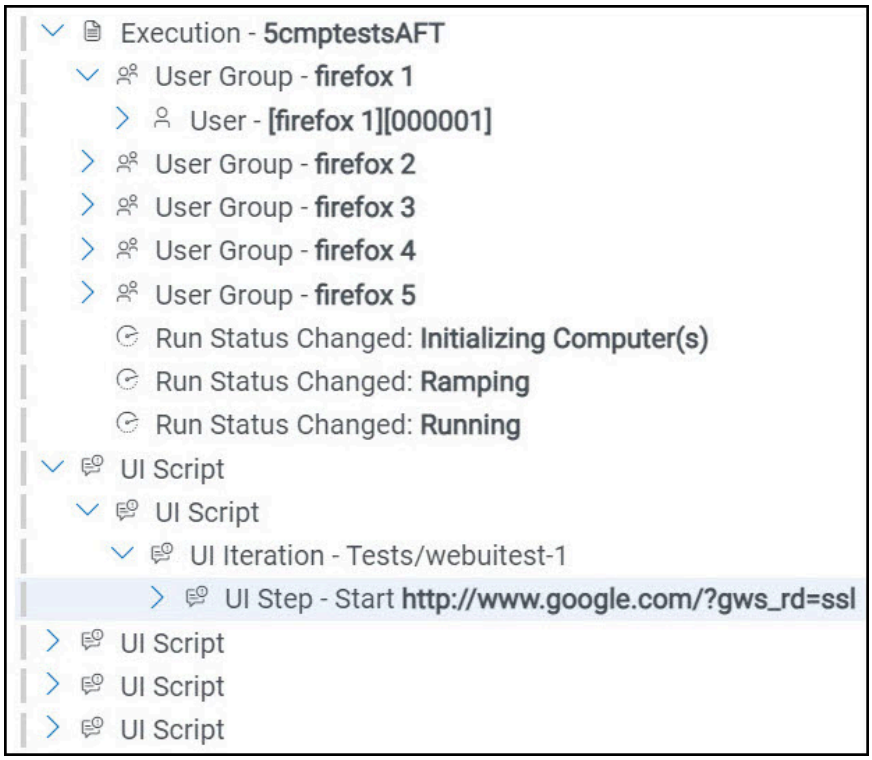
Details about...	Examples as displayed in the Test log page...
	May 23rd 2022 4:38 PM
The activities and events captured in the test log are displayed in the **Tree view** in the **Test log** panel.	An example of the log in the tree view that is displayed is as follows: 
An event in the test log. Details of the event is displayed in the **Properties** panel.	An example of the details that is displayed in the **Properties** panel is as follows:

Table 620. Test log details (continued)

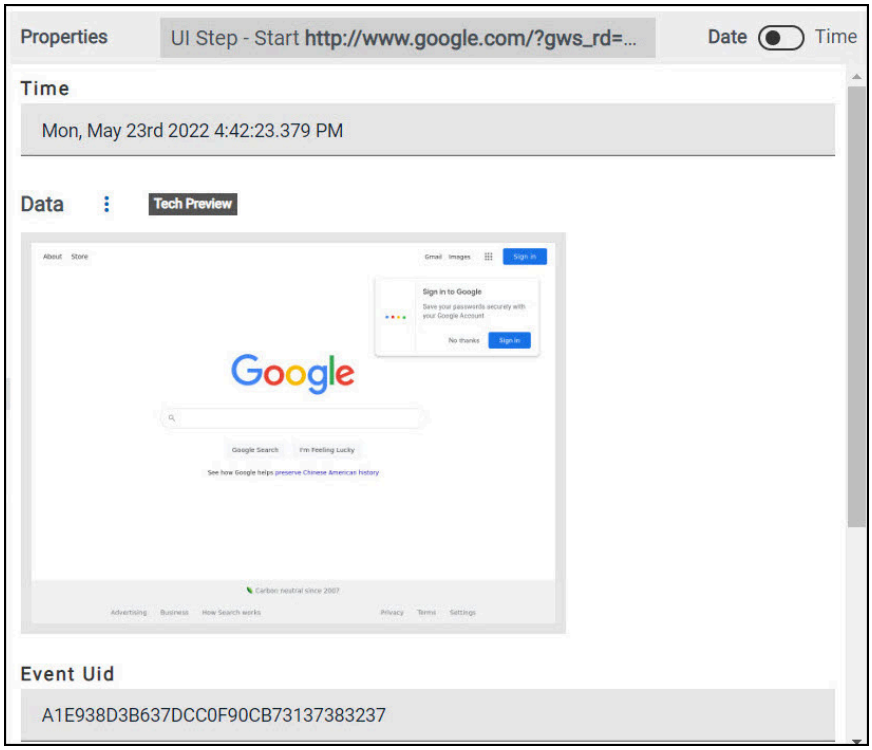
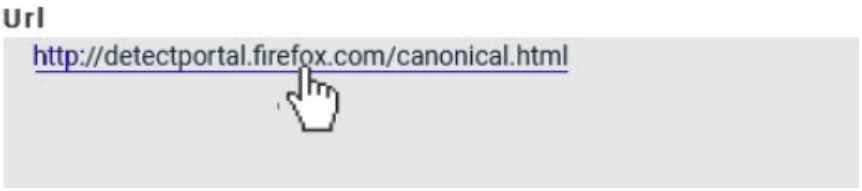
Details about...	Examples as displayed in the Test log page...
	
The link to a URL that is recorded in an event. The URL is displayed as a link. When you click this link, the specific URL opens in a web browser.	An example of a link recorded in the event. 
An attachment of an event in the test log. The details of the attachment is displayed in the **Properties** panel.	An example of a text extracted from the attachment.

Table 620. Test log details (continued)

Details about...	Examples as displayed in the Test log page...
The reason of a Verification Point (VP) that is captured in an event in the test log is displayed in the **Properties** panel.	An example of the reason in a VP that is displayed in the **Properties** panel is as follows:

You can perform the actions detailed in the following table to change the view of the test log, find details of the events captured, navigate through the log, view or download attachments that are captured.

Table 621. Tasks that you can perform on the Test log page

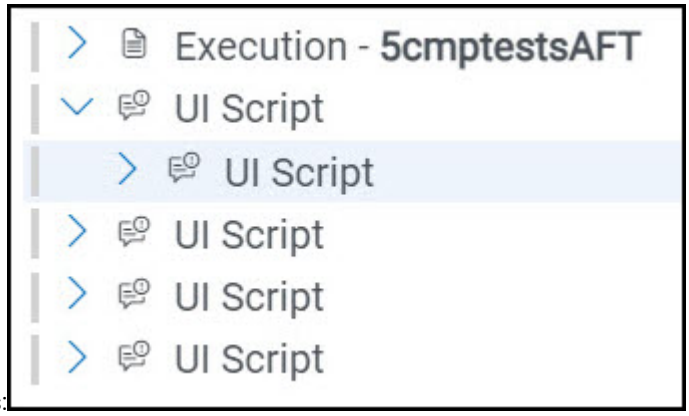
Task	Action
Changing the *Tree view* to a *Flat view* for the display of events in the test log panel. **Note:** The *Tree view* is the default view of the events in the test log.	Click to toggle from the **Tree view** to the **Flat view**. **Note:** You can change the view from the one that is displayed by clicking .
Expanding the display of events in the test log panel.	Click to expand the events that are contained within the event selected in the test log panel. For example, the collapsed view in the test log panel can be as follows: 

Table 621. Tasks that you can perform on the Test log page (continued)

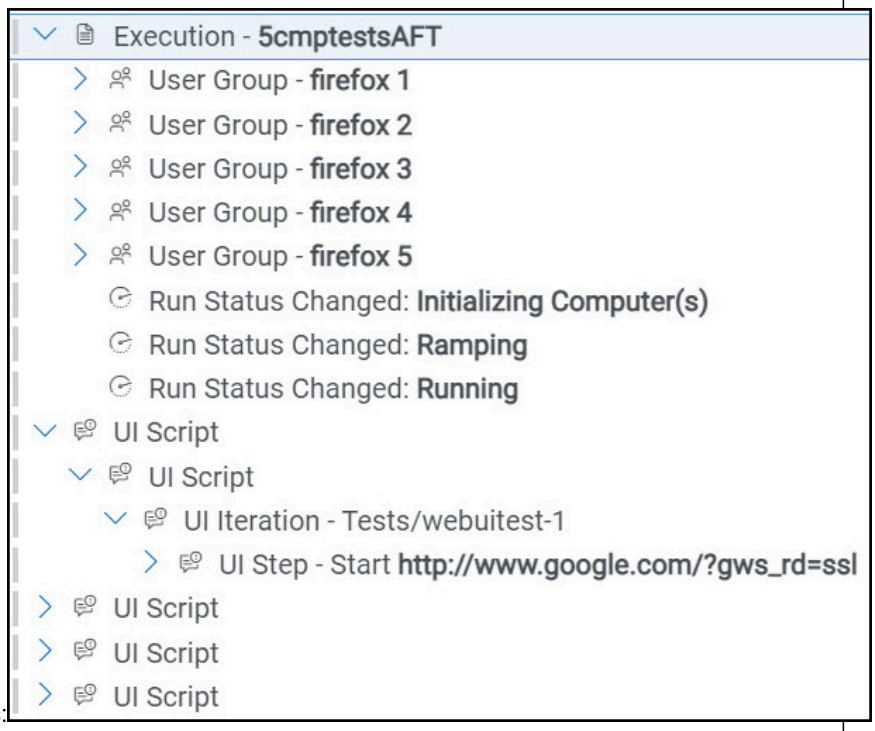
Task	Action
	When the event is expanded, the view in the test log panel can be as follo  ws:
Collapsing the display of events in the test log panel.	Click Collapse to collapse the events that are contained within the event selected in the test log panel.
Viewing primary activities when all activities are displayed.	Perform the following actions in the test log panel: 1. Click the **Open action menu** icon. 2. Click **Show only primary activities**. The primary activities captured in the test log are displayed.
Viewing all activities when only primary activities are displayed.	Perform the following actions in the test log panel: 1. Click the **Open action menu** icon. 2. Click **Show all activities**. All the activities captured in the test log are displayed.

Table 621. Tasks that you can perform on the Test log page (continued)

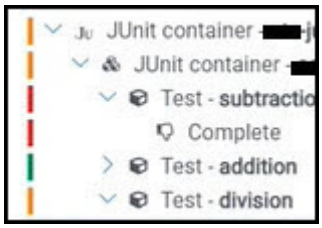
Task	Action
Viewing failures captured in the test log.	The test log displays events marked with a colored vertical bar as follows: • A green bar is marked for an event that passed in the test step. • An orange bar is marked for an event that had an error in the test step. • A red bar is marked for an event that failed in the test step. For example, the view can be as follows:  You can navigate up and down in the view from one error or failed event to another one by using arrows .
Searching for events in the test log.	In the search bar, you can enter text to find instances of events and properties that match with the text you entered. As a result of the search, the title of the nodes in the view and the names of the properties are highlighted. When the number of results found is displayed, you can press the **F3** key to navigate from one search result to another in the tree view or flat view. You can press **Ctrl** and **F3** to navigate from one result to another in the properties pane.
Viewing the properties related to the event that you select in the test log panel.	When you select an even in the test log panel, its properties are displayed in the properties panel. For example, the details that are displayed for an event can be as follows:

Table 621. Tasks that you can perform on the Test log page (continued)

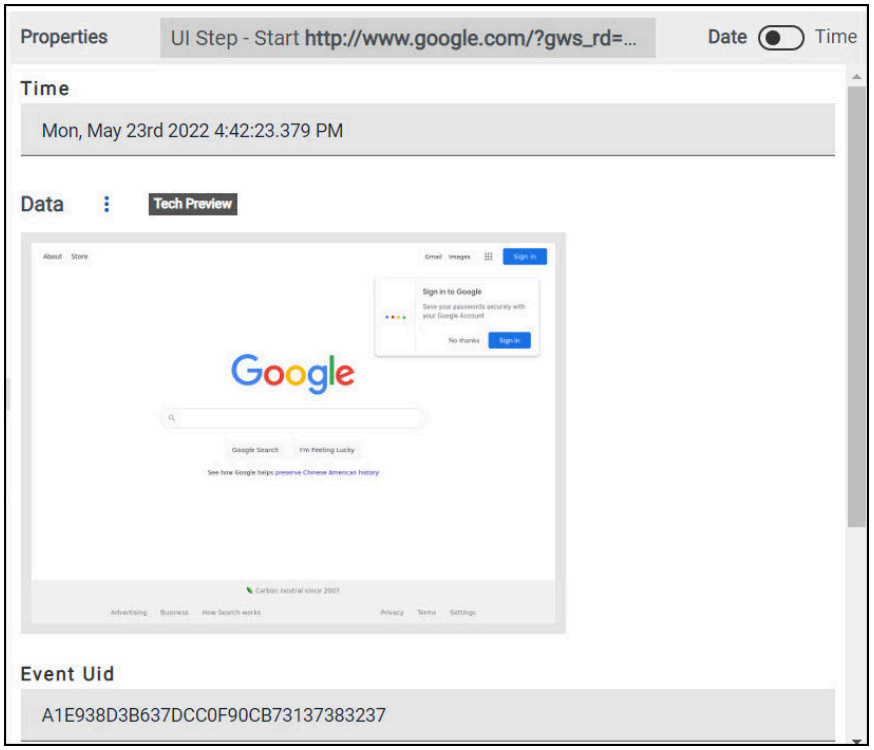
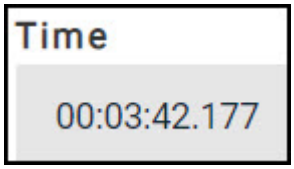
Task	Action
	 You can view the following information: • The variable in the event is displayed. • The date stamp showing the time at which the event was captured in the test log. You can toggle between the calendar time and the duration from the time the test started its run in the cluster by clicking the **Date** and **Time** toggle icon . In the example, the date that is displayed in the **Time** field is as follows: When the **Time** option is selected, the duration that is displayed in the **Time** field is as follows:

Table 621. Tasks that you can perform on the Test log page (continued)

Task	Action
	 This time is the time that has elapsed from the start of the test run to the event in the test step. • The name of the event. • The event ID. • Data that was captured as part of the test step. The items that are captured can be images, text files, PDF files, XML files, JSON files, and other supported formats. You can view the attachment that is in the **Data** field, if it is supported to be viewed in a web browser by clicking the attachment. The attachment displays in a browser tab other than the one in which the **Test log** page is displayed. You can also copy the URL of the attachment or download a copy to your computer to use the attachment elsewhere.
Viewing all test logs in the **Test log** page.	When you view the test log of a specific test asset and you want to return to the **Test log** page to view the test logs of the other test assets, you can click the **Back to test logs** option.

Test log queries

When a test run is complete in Test , if a test log is generated, you can run queries to extract and view test log data. The query is used to search for the events or activities in the test logs with a project or a result scope.

The query page contains a configuration section in which you enter and select search criteria.

The difference between an activity and an event are the following ones:

- An activity is characterized by a duration whereas an event is characterized by a particular time.
- An activity contains events.

When you define the search criteria and run a query, the returned test log data is displayed in a table. You can manage the data in the table by adding, moving, or deleting the columns.

Prerequisites

To get result log data queries, you must configure your tests with one of the following parameters to enable the test log and data query process:

- For JMeter, JUnit, Postman and HCL AppScan CodeSweep Suites, set the **history cisterna** parameter in the **Additional Configuration Parameters** field in the **Advanced** settings

.
- For API tests and Suites, AFT Suites and Functional test, and Compound Tests, set the **-history cisterna** parameter in the **Program Arguments** field in the **Advanced** settings.

Note: When you set this parameter, you can run your queries to collect test log data into Test.

Query scope

A **Test result query** link is available in the test reports in the **Results** page when a test log is generated. If you click the link, it opens the query page in Test where you can view all the events or all the activities for the current test result only. In the **Scope** field of the query page, you find the result id. You can run new queries with different properties as search criteria for this result.

You can also run new queries with different properties as search criteria for all the results in a project if you open the query page from **Analyze > Queries** in the navigation pane. In this case, the **Scope** field is at the project level.

In the **Queries** page, you can perform the following actions:

- Filter the events or activities.
- Sort your query results in an ascending or a descending order.
- Add, move, or delete fields in the columns of the table that displays the data of the query.
- Save a query, rename, reset or delete a query.
- Export a query.

You can export the queries to Shell, Curl, or PowerShell scripts. You can select the script, the output format, and the number of results that are returned by the query.

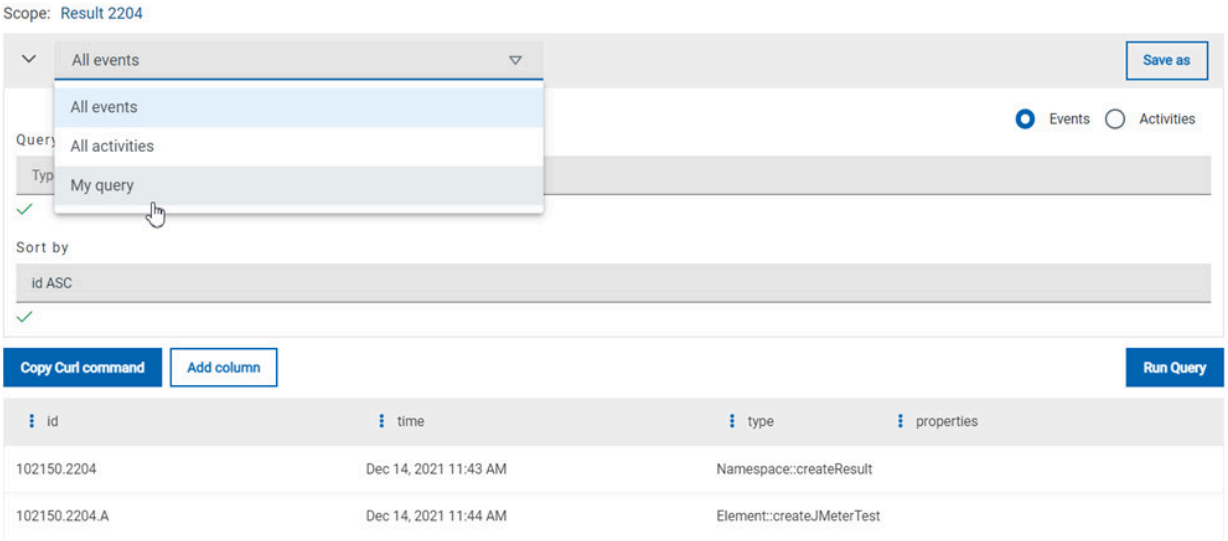
Configuring and running log queries

When a test run is complete and a test log created, you can configure and run queries to view test log data in Test for one test log result or for all the test log results in a project.

Before you begin

You must have completed the following tasks:

- Configured and run a test. See Test log queries on page 1538.
- Opened the Queries page from the Results page or the navigation panel.



About this task

In the **Queries** page, you can configure a query and save it. You can customize the way the log data is displayed in the results table.

1. Click **Events** or **Activites**.
2. Enter properties and operators in the **Query filters** to filter the list of events or activities.

   Note: You can use the **Ctrl-Space** keyboard shortcut in this field to get the list of properties and operators that you can enter as filters to create your query.

   Example: Select **Activities** and enter 'type is JUnitTest' in **Query filters**.

3. Click **Run Query**.
4. Perform the following steps to save a query:

   a. Click **Save as**.
   b. Enter a name to identify the new query and click **Save**.

   Example: Enter 'My query'.

   Result

   The new query is displayed in the events and activities list.
5. Perform any of the following actions to configure the results table:

- Enter a property followed by the **ASC** operator or **DESC** in **Sort by** to sort the data in the table.

  Note: ASC is used to sort the data from the lowest value to the highest value and **DESC** to filter the data in a descending order. You can use the **Ctrl-Space** keyboard shortcut to assist you. If you enter a combination of sort criteria, separate them with commas.

  Example: Click the **Sort by** field, press **Ctrl-Space** and select in the list 'time' and enter 'ASC'.

- Click the action menu in a column of the table to change the column display, and then perform any of the following actions:
  - Click **Insert new column**, enter a field name in **Search new column** and click **Add**.
  - Click **Move** to move a column before or after another one in the table.
  - Click **Remove** to remove a column from the table.
- Follow these steps to add a column to the end of the table:
  a. Click **Add column**.

     Result: A window opens.

  b. Enter a field name in **Search new column**.
  c. Click **Add**.

6. Click **Save** to save the configuration of the results table.
7. Follow these procedures to manage your query after it is created:

   - Click the action menu that is next to the **Save** button and select **Reset** if you have just modified a query and you want to undo your modification.
   - Click the action menu and select **Delete** to delete the new query from the **Events and activities** list.
   - Click **Copy Curl command** if you want to export a query.

     Note: The curl command is copied to the clipboard. You can paste it in a command interface to use the query in an external tool.

Adding additional properties in test results

When you want to add additional properties to be included in test results after the test is run, you can do so by configuring a webhook to receive the customized results and defining the parameters in Test .

Before you begin

You must have completed the following tasks:

- Read and understood Configuration of a webhook on page 1762.
- Read and understood Configuring a webhook template on page 1763 and Configuring a server webhook on page 1771.

About this task

You can set up webhooks to send notifications to various messaging apps such as Microsoft Teams or Slack, or to any application that can receive an HTTP POST request, whenever specific events happen on Test. For instance, a webhook can be triggered when a test run fails. After the event, the payload that is received by the application contains the details of the event, including the additional parameters that are added. If you want the result events to be correlated when using systems such as DevOps Velocity, you must add the properties that are present in the webhook template as additional parameters in the **Execute test asset** window.

1. Navigate to your project in Test.
   The **Overview** page of the project is displayed.
2. Click **Manage > Webhooks**.
   The **Templates** tab is displayed.
3. Click **New Template**.
   The **New template** page is displayed.
4. Perform the following actions on the **DETAILS** tab.
   a. Enter a unique name for the template.
   b. Create a new channel type by clicking **Create new channel type ...** option.

      The **New channel type** dialog is displayed.

   c. Enter a unique name for the channel type, and then enter a description for the channel type.
5. Click **Apply**.
   The channel type is created and added to the **Channel Type** list.
6. Click **This template is suitable for events** and select the events from the **Execution Events** list.
7. Click the **Template** tab.
8. Click one of the following icons to select a predefined template sample in the **Template body**:
   - The **infinity** icon to insert a DevOps Velocity template
   - The **cloud** icon to insert a cloud event JSON template
   - The **Microsoft Teams** icon to replace a template with a Microsoft Teams template
   - The **Slack** icon to replace a template with a Slack template

   A message is displayed about the action that inserts a text in the template body.
9. Click **Ok**.
10. The text is inserted in the **Template body**.
11. Click **Apply**.
    The template is created and added to the **Templates** tab.
12. Click the **Webhooks** tab, and then click **New Webhook**.
    The **New Webhook** page is displayed.

13. Enter a name for the webhook in the **Name** field.
14. Enter the callback URL of the application or channel for which you are configuring a webhook in the **Webhook URL** field.
15. Click the **Channel Type**, and then select the channel type that you just created from the list.
16. Enter a description for the webhook in **Description**.
17. Select the template that you just created from the drop-down list.
18. Click **Apply**.
    The webhook is created and added to the **Webhooks** tab.
19. Click **Execution**, in the navigation pane.
20. Select the test that you want to run and click in the **Actions** column to open the **Execute test asset** window.
21. Click **Advanced settings**.
22. Enter a parameter name that starts with result.property in the first edit field under **Additional Configuration Parameters**.
    For example, result.property.test_version.
23. Enter a value for the newly added parameter in the second edit field under **Additional Configuration Parameters**. For example, v1.5.14.
24. Click **Add** for the added parameter.
    You can add multiple parameters.
25. Click **Execute**.

Results

The additional parameter values are captured while your test is running. You can see the values in the response notification that you get for the webhook after the test run is complete.

What to do next

Navigate to the application or channel for which you have configured the webhook for receiving test result notifications, and then view the results.

Related information

Prerequisites to running tests on page 1057

Test run configurations on page 1134

Machine learning capabilities in DevOps Test

Test uses Machine learning (ML) algorithms to analyze tests that match certain requirements and criteria, and then presents the findings as insights and recommendations after the tests are run. You can use the insights and recommendations to interpret the test results and identify the problem areas in the system under test.

Test has incorporated ML capabilities to provide an automated means to identify problems in the system under test.

Tests analyzed through ML

The following types of tests are supported for analysis by using ML capabilities in Test:

- Performance tests
- Schedules

Parameters analyzed through ML

The following parameters in Performance tests or Schedules are supported for analysis by using ML capabilities in Test:

- The **Response Time Lock-Step Pattern** parameter.
- The **Response Time Standard Deviation Pattern** parameter.
- The **Throughput Drop Pattern** parameter.
- The **Performance Agent Health** parameter.

Test uses different analyzers for each of the parameters. For example, Test uses the **Response Time Lock-Step Pattern** analyzer to analyze the tests for the **Response Time Lock-Step Pattern** parameter.

Criteria for analysis of the parameters

The following table lists the criteria used by the analyzers for analysis of the parameters in Test:

Parameter analyzed	Criteria for analysis
Response Time Lock-Step Pattern	The analyzer identifies the **Response Time Lock-Step Pattern** parameter in the overall page response time observed against the user count based on the following criteria: • Tests have a minimum of 20 users. • Tests run for longer durations for a trend to be observed. • Tests contain a minimum of three different user groups and each group has an activity of more than two minutes.
Response Time Standard Deviation Pattern	The analyzer attempts to detect the response time of pages that are more than thrice the value of the standard deviation calculated for the page response time as the **Response Time Standard Deviation Pattern** parameter.
Throughput Drop Pattern	The analyzer attempts to detect sudden drops in network throughput as the **Throughput Drop Pattern** parameter that is based on the following criteria:

Parameter analyzed	Criteria for analysis
	Note: Sudden drops in throughput might be related to the performance tool itself, issues with network connectivity, or issues with signal-scalability of the system under test. • Tests have a minimum of 20 users. • Tests have stages with a reasonably high number of samples within a stage. • The length and intensity of the throughput drops that are less than 20% of the median observed within a stage are considered by the analyzer. When no throughput is received within 60 seconds, such drops are considered in the analysis. Short shark-tooth patterns in drops are not considered by the analyzer. • Parts of the stage with increasing users are not considered for analysis. For example, when the test adds one user every minute and the user range is between one user to 500 users, the **Throughput Drop Pattern** parameter is not analyzed.
Performance Agent Health	The analyzer inspects the health of agents that are participating in a performance execution and identifies potential configuration issues based on the following criteria: • **Minimum requirements** ◦ At least 10 users ◦ At least 10-minute run time ◦ Requires schedule with at least 1 static agent (Performance Agent Health insight is not generated if you are running the test on local host) • **Blocking threads** ◦ Considered as Blocking threads when 90% of available threads are used for 20% of the run time • **Pegged CPU** ◦ Considered as Pegged CPU when 90% of CPU is utilized for 10% of the run time • **Pegged CPU for interval** ◦ Considered as Pegged CPU for interval when there are intervals of high CPU utilization greater than 95% For example, above 95% of CPU utilized for 50% of a 10-minute interval or 98% of CPU utilized over a shorter interval. ◦ Considered as Pegged CPU for interval when there are spikes in CPU utilization during long test runs

Parameter analyzed	Criteria for analysis
	The agents that are not configured properly can distort performance assessments of the system under test. The analyzer can provide multiple insight types. The insight recommendation texts are parametrized and they provide specific recommendations such as identifying the agents with issues. You can view the following insights: • Pegged CPU for run insight • Pegged CPU for interval insight • Blocked thread insight

Analysis at the project level

When you as a *Project Owner* or *Tester* configure a run of a Performance Test or a Schedule from your project in Test, you must ensure that the criteria required for ML analysis are met. During the test run, Test uses its ML capabilities to analyze the supported parameters and presents its findings as insights on the **Insights** page.

Any member of your project can view the insights by clicking **Analyze > Insights** from the navigation in the project or from the **Insights** section on the **Overview** page in the project.

Project members after viewing the insights can react by either agreeing or disagreeing with the insights suggested by Test.

Analysers in a team space

You can view the list of analysers and description about the analyzers available in a Team space by clicking **Infrastructure > Analyzers** from the navigation in the team space. Also, you can enable or disable the analyzer, view the count of the reviewed insights and the review history of the insights generated by the analyzers.

After viewing the recommendation for a specific parameter, members of the project can alter the default threshold value that specifies the level of confidence required in the analysis of the parameter and run the tests again to improve the accuracy of the recommendations and insights.

You can also improve the accuracy of the recommendations by running multiple tests with differing loads repeatedly.

Using Machine learning capabilities

You can find information in the following documentation about using Machine learning (ML) capabilities in Test:

- To understand the requirements of Performance tests or Schedules that are analyzed by using ML capabilities in Test, see Test run considerations for using Machine learning capabilities on page 1064.
- To configure a run of a Performance Test, see Configuring a run of a Performance test on page 1296.
- To configure a run of a Schedule, see Configuring a run of a Rate Schedule or VU Schedule on page 1404.
- To view the ML insights into the Performance tests or Schedules in your project, which were analyzed after a run, see Viewing Machine learning insights on page 1547. You can also select your reaction to agree or disagree with the insights provided when you view the insight.

- To understand how to interpret the insights, see Interpretation of insights on page 1549.
- To view the ML analysis of the supported parameters when Performance tests or Schedules that are in different projects in your team space are run, see Viewing Machine learning analyzers on page 1553.
- To understand how to interpret the recommendations of the ML analysis of the supported parameters, see Interpretation of parameter analysis on page 1555.
- To view the reactions of project members to the ML insights, see Viewing reactions marked by members to Machine learning insights on page 1558.
- To modify the settings of the ML analyzer for a specific parameter, see Modifying Machine learning analyzer settings on page 1560.
- To delete the ML insights provided about the Performance tests or Schedules that were analyzed, see Deleting Machine learning insights on page 1562.
- To rerun the ML analysis without rerunning the test asset, see Rerunning Machine learning (ML) analysis on page 1563.

Viewing Machine learning insights

When you want to use Machine learning (ML) capabilities in Test , you need to run Performance Tests or Schedules that conform to certain criteria. After the test run is completed, you can view the ML insights on the **Insights** page in your project.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Ran the Performance Test or Schedule that matches the minimum requirements for ML analysis in Test. See Configuring a run of a Compound Test that contains performance tests on page 1260 or Configuring a run of a Rate Schedule or VU Schedule on page 1404.

About this task

If the Performance Tests or Schedules match the requirements and criteria for ML analysis, Test analyzes the test results for the relevant parameter and presents its findings as insights into the parameter.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Perform any of the following actions to view the insights:
   - Scroll down the **Overview** page to see the **Insights** section.

     Go to Step 4 on page 1548 for the actions that you can perform to view the insights.
   - Click **Analyze > Insights** in the navigation pane.

The **Insights** page is displayed.

Go to Step 5 on page 1548 for the actions that you can perform to view the insights.

4. Perform the following steps to view a specific insight from the **Overview** page:

   The **Insights** section on the **Overview** page displays a table of the insights for the supported parameters identified from the tests that were run in the project. The table lists the level of confidence with which the analysis of the specific parameter was done, the parameter analyzed, the name of the test asset, and the timestamp of the completed analysis.

   a. Click any item in the row of a specific insight to view the details of that insight.

      The **Insights** page is displayed with the details of the specific insight.

      Note: If you want to view the list of the other insights that are available in your project, you can click **Back to all insights**, and the list of available insights in the project is displayed in a tabular view.

   b. Go to Step 6 on page 1548.

5. Perform the following steps to view a specific insight from the **Insights** page:

   a. Identify the test asset or the parameter by performing any of the following actions:

      - Search for the test asset or parameter by entering any text contained in its name in the **Search** field.
      - Sort the items in the table of insights by clicking any of the column headers, and then identify the insight.

        For example, you can sort the items based on the name of the test asset in an ascending order or descending order, and then identify the test that you ran.

      You identified the insight.

   b. Click the **Expand** icon > of the insight.

      The details of the specific insight are displayed.

   c. Go to Step 6 on page 1548.

6. Perform the following steps when the details of the insight are displayed.

   a. Read the details about the analysis and the issues that are identified by the analyzer.

   b. Correlate the graph analyzed by the analyzer with your observations on the parameter.

   c. Select your reaction to either agree or disagree with the insight by performing the following actions:

- Click the **Agree** icon, if you agree with the insights provided.

  After you agree, the icon changes to and displays the text `Agreed by username` when you hover over the icon.

- Click the **Disagree** icon, if you disagree with the insights provided.

  After you disagree, the icon changes to and displays the text `Disagreed by username` when you hover over the icon.

Note: A member of a project can only react once by either agreeing or disagreeing with the insight.

Results

You viewed the details of the insights into the parameters that are analyzed by Test.

What to do next

You can perform any of the following tasks:

- Understand how to interpret the insights. See Interpretation of insights on page 1549.
- Delete an insight. See Deleting Machine learning insights on page 1562.
- Rerun the analysis so that the deleted instance of the insight is redisplayed. See Rerunning Machine learning (ML) analysis on page 1563.

Related reference

Interpretation of insights on page 1549

Interpretation of insights

You can find information about interpreting the Machine learning (ML) insights into the supported parameters that are analyzed by Test after certain tests are run.

Parameters analyzed through ML

The following parameters in Performance tests or Schedules are supported for analysis by using ML capabilities in Test:

- The **Response Time Lock-Step Pattern** parameter.
- The **Response Time Standard Deviation Pattern** parameter.
- The **Throughput Drop Pattern** parameter.
- The **Performance Agent Health** parameter.

Test analyzes the parameters based on certain criteria as detailed in Criteria for analysis of the parameters on page 1544 and then presents its findings as insights about the parameters.

Details of the insights

The insights about the parameters analyzed are displayed in the **Insights** section on the project **Overview** page or on the **Insights** page in a project.

The insights analyzed for the parameters from various tests that were run in the project are listed in a table. The table lists the following information:

- Name of the test asset in the **Test** column.
- The parameter analyzed in the **Insight** column.
- The level of confidence with which the analysis of the parameter was done in the **Confidence** column. The level of confidence displayed for the analysis can be as *Lowest*, *Low*, *Medium*, *High*, or *Highest*. You can view the confidence level as a number when you hover the cursor over the text in the **Confidence** column.
- The timestamp of the completed analysis in the **Identified on** column.
- The labels that you added for the specific test run in the **Labels** column.
- The components in the System Model that are associated with the test asset in the **Components** column.

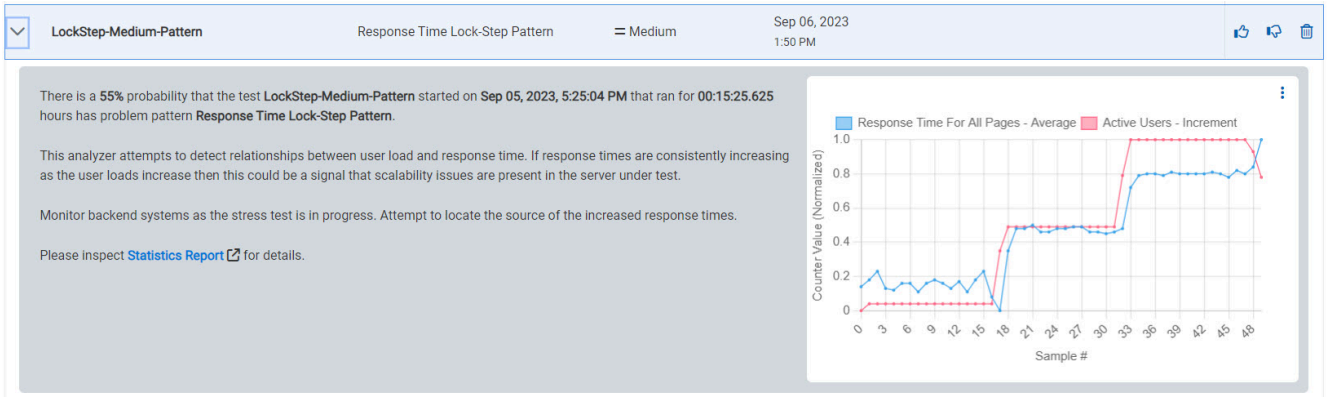
You can click the **Expand** icon to view the details of the insight.

You can find information as an example about the following insights that help you to interpret the insight:

- Insights on the Response Time Lock-Step Pattern parameter on page 1550
- Insights on the Response Time Standard Deviation Pattern parameter on page 1551
- Insights on the Throughput Drop Pattern parameter on page 1551
- Insights on the Performance Agent Health parameter on page 1552

Insights on the Response Time Lock-Step Pattern parameter

When you expand the insight about the **Response Time Lock-Step Pattern** parameter, you can view the details displayed similar to the following sample:

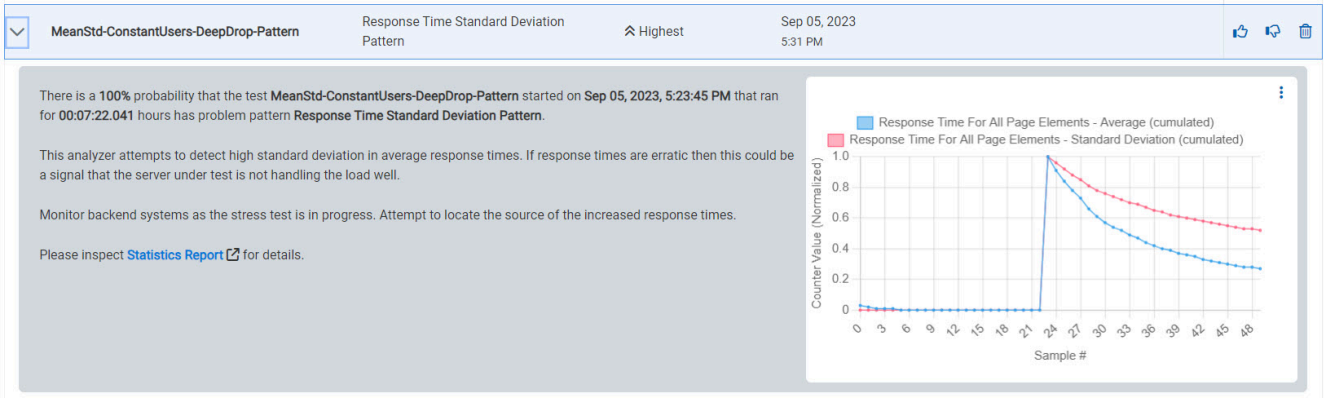


The following information is displayed as insights into the test on analyzing the **Response Time Lock-Step Pattern** parameter:

- The probability of finding a problem in the test.
- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as suggestions.
- The graph of the observed data points.
- The link to the Statistics Report generated for the test.

Insights on the Response Time Standard Deviation Pattern parameter

When you expand the insight about the **Response Time Standard Deviation Pattern** parameter, you can view the details displayed similar to the following sample:

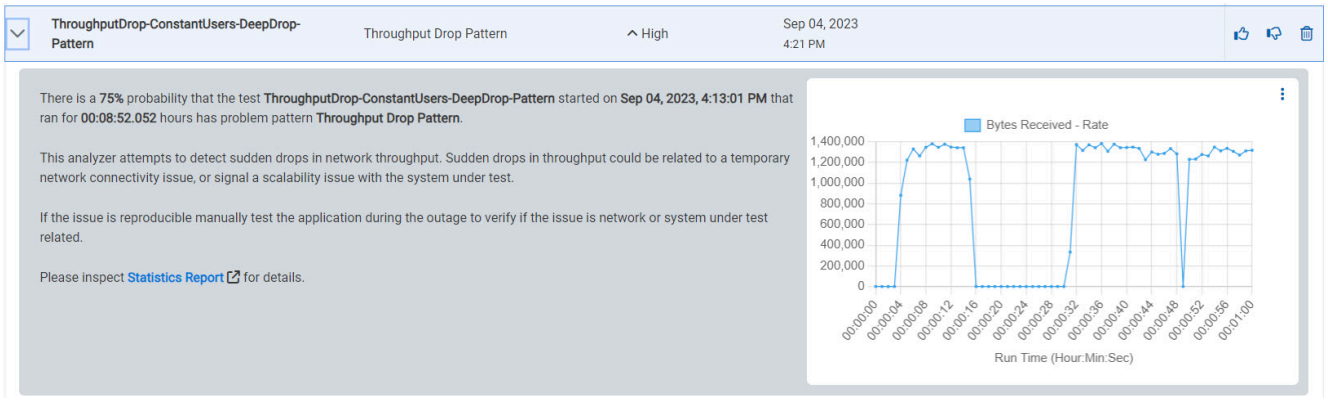


The following information is displayed as insights into the test on analyzing the **Response Time Standard Deviation Pattern** parameter:

- The probability of finding a problem in the test.
- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as suggestions.
- The graph of the observed data points.
- The link to the Statistics Report generated for the test.

Insights on the Throughput Drop Pattern parameter

When you expand the insight about the **Throughput Drop Pattern** parameter, you can view the details displayed similar to the following sample:

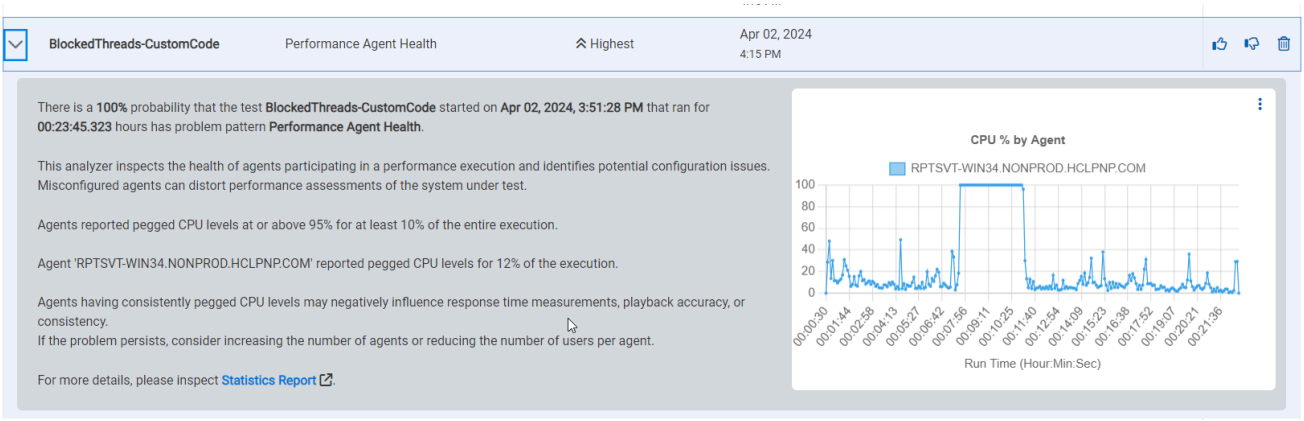


The following information is displayed as insights into the test on analyzing the **Throughput Drop Pattern** parameter:

- The probability of finding a problem in the test.
- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as suggestions.
- The graph of the observed data points.
- The link to the Statistics Report generated for the test.

Insights on the Performance Agent Health parameter

When you expand the insight about the **Performance Agent Health** parameter, you can view the details displayed similar to the following sample:



The following information is displayed as insights into the test on analyzing the **Performance Agent Health** parameter:

- The probability of finding a problem in the test.
- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as suggestions.
- The graph of the observed data points.
- The link to the Statistics Report generated for the test.

Related information

Viewing Machine learning insights on page 1547

Viewing Machine learning analyzers

Test analyzes certain types of tests that are run in a team space by using its Machine learning (ML) capabilities. You can view the list of analysers and description about the analyzers available in a Team space.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Ran the Performance Test or Schedule that matches the minimum requirements for ML analysis in Test. See Configuring a run of a Compound Test that contains performance tests on page 1260 or Configuring a run of a Rate Schedule or VU Schedule on page 1404.

About this task

If the Performance Tests or Schedules that are run from projects in a team space match the requirements and criteria for ML analysis, Test analyzes the tests during their runs.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Analyzers** in the navigation pane.

   The **Analyzers** page is displayed.

   You can view the parameters that are supported for ML analysis listed in a tabular format. The information displayed is as follows:

Column header	Description
Status	Displays the status of the parameter analysis as active or inactive in the team space. The default value of the analysis is set as active or enabled.
Name	Displays the name of the parameters that are supported for ML analysis, which is as follows: ◦ **Response Time Lock-Step Pattern** ◦ **Response Time Standard Deviation Pattern** ◦ **Throughput Drop Pattern** ◦ **Performance Agent Health**
Version	Displays the version of the ML analyzer.
Last Run	Displays the date when the tests were run for the analysis.

<table>
<tr><th>Column header</th><th>Description</th></tr>
<tr><td>Unreviewed insights</td><td>Displays the number of insights into the specific parameter analyzed from different tests that are not reviewed by the project members.</td></tr>
<tr><td>Actions</td><td>Displays the following icons for the actions that you can perform on each parameter analyzed:
<table>
<tr><th>Icon</th><th>Description</th></tr>
<tr><td> </td><td>Displays the Edit Analyzer dialog when clicked. You can modify the analyzer settings for subsequent analysis.</td></tr>
<tr><td> </td><td>Displays a dialog that contains the list of the reactions marked by different members on insights into the specific parameter when clicked. The agreement or disagreement reactions by members with the date and time of marking them are listed in chronological order in the dialog. You can view the insight on the Insights page in the project when you click .</td></tr>
</table>
</td></tr>
</table>

3. Click the **Expand** icon of the parameter to view the details.

   The details of the parameter are displayed.

4. Read the details about the analysis and recommendations for troubleshooting or resolving the problems detected in the analysis.

   See the information in the related links to understand how to interpret the analysis of the parameters.

Results

You viewed the details of the parameters that are analyzed by Test in tests that were run in the team space.

What to do next

You can perform any of the following tasks:

- Understand how to interpret the recommendations from the analysis of the parameters. See Interpretation of parameter analysis on page 1555.
- View reactions by members to an insight. See Viewing reactions marked by members to Machine learning insights on page 1558.
- Modify the ML analyzer settings. See Modifying Machine learning analyzer settings on page 1560.

Related reference

Interpretation of parameter analysis on page 1555

Interpretation of parameter analysis

You can find information about interpreting the details of each of the Machine Learning (ML) analyzers that analyze the parameters after certain tests are run in Test .

Parameters analyzed through ML

The following parameters in Performance tests or Schedules are supported for analysis by using ML capabilities in Test:

- The **Response Time Lock-Step Pattern** parameter.
- The **Response Time Standard Deviation Pattern** parameter.
- The **Throughput Drop Pattern** parameter.
- The **Performance Agent Health** parameter.

Test analyzes the parameters based on certain criteria as detailed in Criteria for analysis of the parameters on page 1544. The ML analyzers analyze the results of tests run in all projects in a team space for each of the parameters. The analyzers present their findings on each of the parameters as an insight on the **Insights** page in the project.

Details of the parameters analyzed

The details about the parameters analyzed are displayed on the **Analyzers** page in the team space.

The parameters are listed in a table with the following information:

Column header	Description
Status	Displays the status of the parameter analysis as active or inactive in the team space. The default value of the analysis is set as active or enabled.
Name	Displays the name of the parameters that are supported for ML analysis, which is as follows: • **Response Time Lock-Step Pattern** • **Response Time Standard Deviation Pattern** • **Throughput Drop Pattern** • **Performance Agent Health**
Version	Displays the version of the ML analyzer.
Last Run	Displays the date when the tests were run for the analysis.
Unreviewed insights	Displays the number of insights into the specific parameter analyzed from different tests that are not reviewed by the project members.
Actions	Displays the following icons for the actions that you can perform on each parameter analyzed:

Column header	Description
	<table><tr><th>Icon</th><th>Description</th></tr><tr><td> </td><td>Displays the **Edit Analyzer** dialog when clicked. You can modify the analyzer settings for subsequent analysis.</td></tr><tr><td> </td><td>Displays a dialog that contains the list of the reactions marked by different members on insights into the specific parameter when clicked. The agreement or disagreement reactions by members with the date and time of marking them are listed in chronological order in the dialog. You can view the insight on the **Insights** page in the project when you click .</td></tr></table>

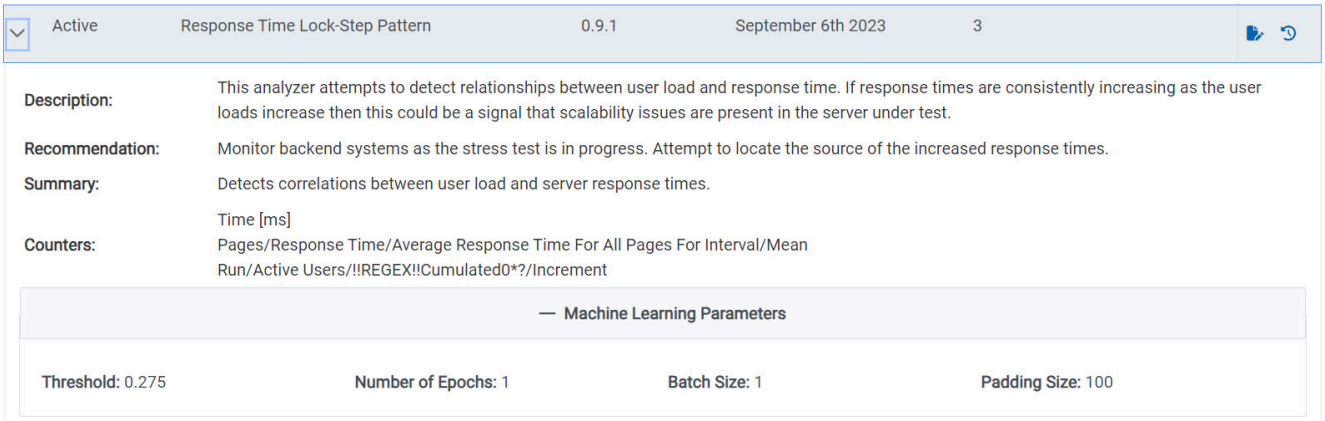
You can click the **Expand** icon to view the details of the parameter analyzed.

You can find information as an example about the following parameters that help you to interpret the parameter analysis:

- The Response Time Lock-Step Pattern parameter analyzer on page 1556
- The Response Time Standard Deviation Pattern parameter analyzer on page 1557
- The Throughput Drop Pattern parameter analyzer on page 1557

The Response Time Lock-Step Pattern parameter analyzer

When you expand the **Response Time Lock-Step Pattern** parameter, you can view the details displayed similar to the following sample:



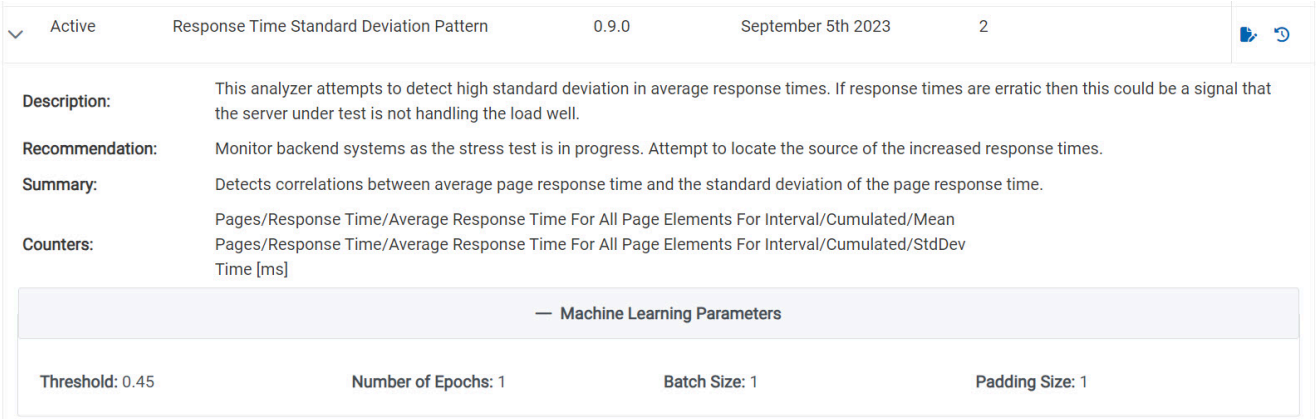
The following information is displayed for the **Response Time Lock-Step Pattern** parameter:

- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as a recommendation.
- The summary of the analysis.

- The counters used in the analysis.
- The parameters used in the analysis.

The Response Time Standard Deviation Pattern parameter analyzer

When you expand the **Response Time Standard Deviation Pattern** parameter, you can view the details displayed similar to the following sample:

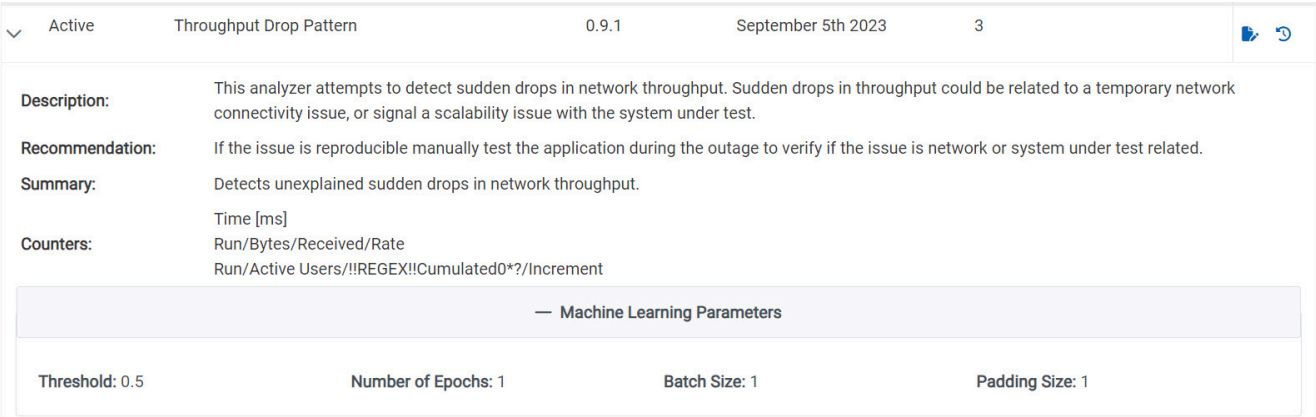


The following information is displayed for the **Response Time Standard Deviation Pattern** parameter:

- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as a recommendation.
- The summary of the analysis.
- The counters used in the analysis.
- The parameters used in the analysis.

The Throughput Drop Pattern parameter analyzer

When you expand the **Throughput Drop Pattern** parameter, you can view the details displayed similar to the following sample:

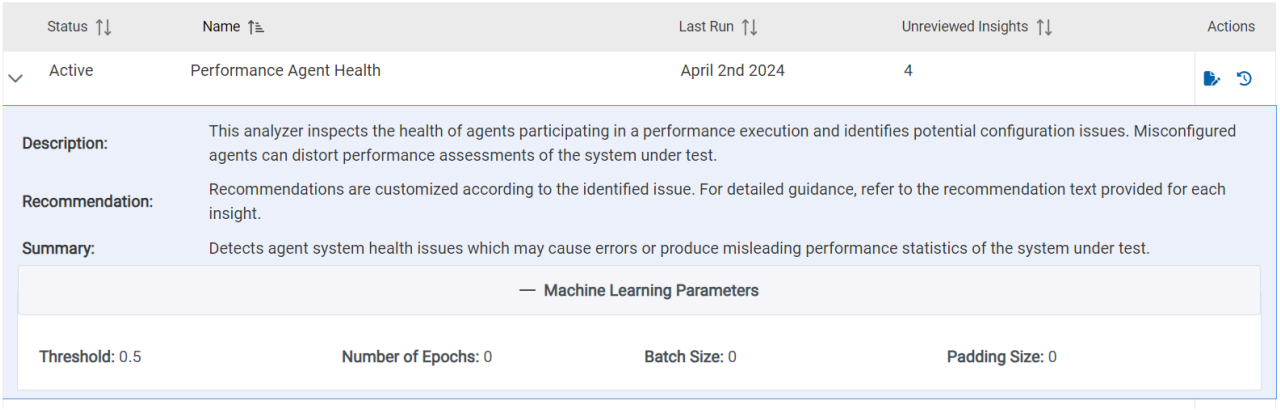


The following information is displayed for the **Throughput Drop Pattern** parameter:

- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as a recommendation.
- The summary of the analysis.
- The counters used in the analysis.
- The parameters used in the analysis.

The Performance Agent Health parameter analyzer

When you expand the **Performance Agent Health** parameter, you can view the details displayed similar to the following sample:



The following information is displayed for the **Performance Agent Health** parameter:

- The description of how the ML analyzer has attempted to analyze the parameter in the test.
- The methods to control the problems are provided as a recommendation.
- The summary of the analysis.
- The parameters used in the analysis.

Related information

Viewing Machine learning analyzers on page 1553

Viewing reactions marked by members to Machine learning insights

You can view the reactions that members of a project mark to show their agreement or disagreement with the insights into the parameters on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Ran the Performance Test or Schedule that matches the minimum requirements for ML analysis in Test. See Configuring a run of a Compound Test that contains performance tests on page 1260 or Configuring a run of a Rate Schedule or VU Schedule on page 1404.
- Viewed the ML insights and marked your reaction to an insight. See Viewing Machine learning insights on page 1547.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Insights** in the navigation pane.

   The **Insights** page is displayed.
4. Identify the test asset or the parameter by performing any of the following actions:

   - Search for the test asset or parameter by entering any text contained in its name in the **Search** field.
   - Sort the items in the table of insights by clicking any of the column headers, and then identifying the insight.

     For example, you can sort the items based on the name of the test asset in an ascending order or descending order, and then identify the test that you ran.

   You identified the insight.
5. Use the information in the following table to interpret the reactions marked for the insight:

If the icon is displayed as	Interpretation of the reaction
	None of the members agreed with the insight.
	At least one member agreed with the insight. **Note:** The name of the member who reacted is displayed as `Agreed by` *`username`* when you hover over the icon.
	None of the members disagreed with the insight.
	At least one member disagreed with the insight. **Note:** The name of the member who reacted is displayed as `Disagreed by` *`username`* when you hover over the icon.

Results

You viewed the member reactions to an insight from the **Insights** page.

What to do next

You can perform any of the following tasks:

- Modify the threshold value of a specific parameter based on the number of disagreements marked by members so that you can improve the accuracy of the insight. See Modifying Machine learning analyzer settings on page 1560.
- Repeat the test runs with the updated thresholds to improve the Machine Learning insights for the specific parameter. See Test run configurations on page 1134.

Modifying Machine learning analyzer settings

When you want to change the decision threshold set for a Machine learning (ML) analyzer or disable the analyzer, you can modify the settings from the **Analyzer** page.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Ran the Performance Test or Schedule that matches the minimum requirements for ML analysis in Test. See Configuring a run of a Compound Test that contains performance tests on page 1260 or Configuring a run of a Rate Schedule or VU Schedule on page 1404.
- Viewed the insights into tests provided by ML analysis. See Viewing Machine learning insights on page 1547.
- Viewed the reactions from members of your project to the ML insight. See Viewing reactions marked by members to Machine learning insights on page 1558.
- Viewed the history of member reactions for a specific parameter analyzer from the **Analyzers** page. See Viewing Machine learning analyzers on page 1553.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Infrastructure > Analyzers** in the navigation pane.

   The **Analyzers** page is displayed. You can view the parameters that are supported for ML analysis listed in a tabular format.

3. Identify the parameter whose analyzer you want to modify, and then click the **Edit Analyzer** icon in the **Actions** column in the row of the parameter.

   The **Edit Analyzer** dialog is displayed.

4. Perform any of the following actions to modify the default settings for the ML analysis of the specific parameter.

If you want to...	Action
Disable the analysis for the specific parameter.	Perform the following steps: a. Click **Is Enabled** to switch the analysis from being enabled or active to become disabled or inactive. **Note:** If you modify the default setting to be disabled, Test does not perform ML analysis of the specific parameter when tests are run subsequently. b. Click **Submit**.
Enable the analysis for the specific parameter, if it is disabled.	Perform the following steps: a. Click **Is Enabled** to switch the analysis from being disabled or inactive to become enabled or active. b. Click **Submit**.
Increase or decrease the threshold level.	Perform the following steps: a. Edit the value in the **Decision Threshold** field and enter a value as required. The threshold value determines the level after which the ML analysis identifies an insight into the specific parameter when the test runs. **Note:** You can retain the default values for the decision threshold initially until a large number of tests are run that contribute to the insights for a specific parameter. You can change the value to a higher threshold when there are a large number of records learned by the ML analyzers. b. Click **Submit**.

Results

You either disabled or enabled the analyzer or changed the decision threshold set for a ML analyzer.

What to do next

You can perform any of the following tasks:

- Rerun the analysis of the test with the changed threshold values without running the tests again. See Rerunning Machine learning (ML) analysis on page 1563.
- Run tests if you changed the decision threshold values so that the ML analysis uses the changed values. See Test run configurations on page 1134.

- View insights into the specific parameter whose decision threshold was modified after the test runs are completed. See Viewing Machine learning insights on page 1547.
- View the ML recommendations for the specific parameter whose decision threshold was modified. See Viewing Machine learning analyzers on page 1553.

Deleting Machine learning insights

After you view the Machine learning (ML) insights into the tests, you might want to delete the insight if many members of your project disagree with the insight.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.
- Viewed the insights into tests provided by ML analysis. See Viewing Machine learning insights on page 1547.
- Viewed the reactions from members of your project to the ML insight. See Viewing reactions marked by members to Machine learning insights on page 1558.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Perform any of the following actions to view and identify the insight:
   - Scroll down the **Overview** page to see the **Insights** section.

     The **Insights** section on the **Overview** page displays a table of the insights for the supported parameters identified from the tests that were run in the project. You can identify the insight and click any item in the row of the identified insight to display the insight on the **Insights** page.
   - Click **Analyze > Insights** in the navigation pane. The **Insights** page is displayed. You can identify the insight that you want to delete.
4. Click the **Delete** icon from the **Actions** column that is in the row of the identified insight.

   The **Delete Insight** dialog is displayed.
5. Click **Yes**.

   The insight is deleted and a confirmation message is displayed.

Results

You deleted an insight into the test arrived at by ML analysis of the test.

What to do next

You can perform any of the following tasks:

- Run tests to let the ML analyzers reanalyze and present the insights. See Test run configurations on page 1134.
- Rerun the analysis without running the tests again so that the deleted instance of the insight is redisplayed. See Rerunning Machine learning (ML) analysis on page 1563.

Rerunning Machine learning (ML) analysis

If you change the decision threshold value set for a parameter analyzer, you can then rerun the analyzer to analyze the parameter for the changed threshold and present the insight without running the tests again.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results** from the navigation pane in your project.

   The **Results** page is displayed.
4. Click the **SUITES** tab, if you ran a Schedule or the **TESTS** tab, if you ran a Performance Test.
5. Identify the test result for which you want to rerun the analysis by performing any of the following actions:
   - Search for the test result by entering any text contained in the test asset name in the **Search** field.
   - Sort the items in the table of results by clicking the **Started by** column header, and then identifying the test runs that you started.
6. Click the **Open action menu** icon from the **Actions** column that is in the row of the identified test result.
7. Click **Rerun analysis** .

   The parameter is analyzed again with the changed threshold vale and the insight presented.

Results

You started the ML analysis of the previously analyzed test result so that the insight can be displayed based on the changed threshold value without running the test again.

Resource monitoring

When you apply load to a system under test, the amount of resources consumed by your system is increasing. If the capacity of the resources does not match the load, you can see performance issues in the results. Resource monitoring in Test helps you monitoring the resources of a system and establish the performance metrics and usage data of the system. Thus, you can observe the health of these resources while a schedule is running.

You can use resource monitoring while running a schedule to capture data, such as processor or memory usage, or to monitor the availability of hosts and services by using counters. Test provides a comprehensive view of a system under test

to help determine problems. Hosts and services can be servers, virtual machines, or any local host or network services. You can also monitor remote hosts and services with agents.

Monitoring a host or a server

You can use resource monitoring to monitor any host and network service. This method is used for host targets that the service can reach directly. With Test, you can also monitor performance metrics that are collected and stored by a monitoring system that is monitoring a host. You can use a Prometheus server or a data collector through an exporter, OpenMetrics exporter, for example.

To monitor resources on a local system, you must add the source of the resource monitoring data from the resource monitoring page. Then you have to enter some connection settings and select the counters to monitor the sources. The metrics and usage data are exposed in a graph for each selected counter.

Monitoring a remote host with monitoring agents

You can monitor remote hosts and services with agents through wider sets of data collectors, computers, and networks. The purpose is to capture CPU load, disk space, memory, and the running process for example.

Agent-based monitoring is useful when remote services are not directly accessible from Test through the network. Agents are closer to the target that you want to monitor. You can set up the agent on an authorized host, for example, when access to an Apache httpd or NGINX server status page or a JVM JMX port, a Prometheus server, or an OpenMetrics exporter is restricted to one or few client hosts only. The configuration task is simplified, and no security changes are required.

Agent-based monitoring is mandatory to collect data from a Windows or Linux Performance host.

To monitor a host across a monitoring agent, you must first set up a resource monitoring agent on the target host for which you want to collect the performance statistics and usage data. The agent establishes a connection with Test. When the connection is set, the agent is showing up in the list of agents on the resource monitoring main page. Then, you can select the agent in the list of target sources in the dialog **Access target from** in the dialog that opens from **Add a Source**.

To learn more about configuring the resource monitoring sources, counters, and agents and collecting and viewing the data inTest, go through the information in the following links.

Resource monitoring capabilities

With Test , you can monitor a host or a service that you add to a project as a source. You can select the statistics counters that are used to collect metrics and the resource usage data throughout the monitoring system. You can visualize metrics and data that are gathered during the monitoring process in Test. You can add the resource monitoring sources and counters in a team space or in a project to collect metrics and resource usage data.

Requirements:

- You must be a *Team Space Owner* to create, modify and remove resource monitoring sources in a team space. Any team space *Member* (not owner) can only view resource monitoring sources.
- You must be a project *Owner* to create, modify and remove resource monitoring sources in a project. Any project *Member* (not the owner) can only view resource monitoring sources.

Supported sources

With Test, you can monitor resources for the following sources:

- **Apache httpd server**: You can optionally use agents.
- **NGINX and NGINX Plus**: You can optionally use agents.
- **Java Virtual Machine**: You can optionally use agents.

  You can monitor JVM resources from a local or a remote system. Some parameters must be set in the command before running the Java Virtual Machine. For more details, see the link to the page about starting a Java Virtual Machine on this page.

- **Windows Performance host**:

  To monitor the performance of a Windows host, you must have installed an agent on the target Windows host

  or on a Windows host that is configured to access the performance data of the Windows host target.

- **Linux Performance host**:

  To monitor the performance of a Linux host, you must have installed an agent on the target Linux host.

- **Prometheus server**:

  You can monitor metrics of a host under test that are collected by a Prometheus server. Prometheus collects metrics from monitored targets by regularly requesting appropriate HTTP endpoints on these targets (called *scraping*).

  The metrics data are collected through data collectors sets (groups of monitoring counters) that are tracking the system performance. The performance data results are stored in the Prometheus database so that they can be consumed by external systems through a REST API.

  In Test, you can query these performance metrics collected by Prometheus servers. 'PromQL' is the language for querying Prometheus metric data. Refer to Prometheus PromQL documentation for information about this query language.

  You can select default queries, or create additional ones.

  A Prometheus server is required. It must be configured to scrap a set of exporters, every 15 seconds by default. Pushgateway, service discovery, and Alertmanager are optional.

If Test cannot directly reach the Prometheus server, you can consider setting up an agent on an authorized host or on the same host that is running Prometheus.

- **OpenMetrics exporter**:

  You can monitor the metrics of a host under test that are collected and exposed through an OpenMetrics format or a Prometheus exporter format.

  Test can scrap metrics exposed by an OpenMetrics or Prometheus exporter through metric counters. No Prometheus server installation is required, you only need to set up one or multiple OpenMetrics, or Prometheus exporters that fit your software or host target.

  If Test cannot directly reach the Prometheus server, you can consider setting up an agent on an authorized host or on the same host that is running Prometheus.

Monitoring an NGINX server

To monitor an NGINX server, you must add the NGINX server source to Test , enter connection settings and select the counters that are used to capture the monitoring usage data.

Before you begin

You must have logged in Test and be the owner of an existing new project.

If you monitor a remote host, it must be connected with the computer that you use to access resource monitoring. See Resource monitoring agents on page 1581.

You can optionally use agents.

About this task

This task applies to NGINX and NGINX Plus.

If you use an agent to connect

1. Click **Infrastructure > Resource Monitoring**.
   The resource monitoring page opens.
2. In the resource monitoring page, click **Add a Source** and select **Add an NGINX server...**.
3. Fill in the following connection settings in the **New server** dialog:

   a. In **Target host**, enter the IP address or hostname and port number of the host where the server to monitor is installed.

   Note: For hosts that are already connected to Test through an agent, you have only the **Access target from** field enabled. Move directly to step 4.

   b. In **Server edition**, select the appropriate server from the drop-down list.

c. In **Path to the status page of the server**, enter the name of the page to view the status of the server.

Note: If you select NGINX Plus (with API version 3) as a source, you must specify the name of the path to view the API root of the NGINX Plus server.

d. In **Security**, select the following options:

Secured with TLS/SSL

If the application server is secured with TLS/SSL.

Trust self-signed certificate

To accept the server certificate.

Do not verify hostname

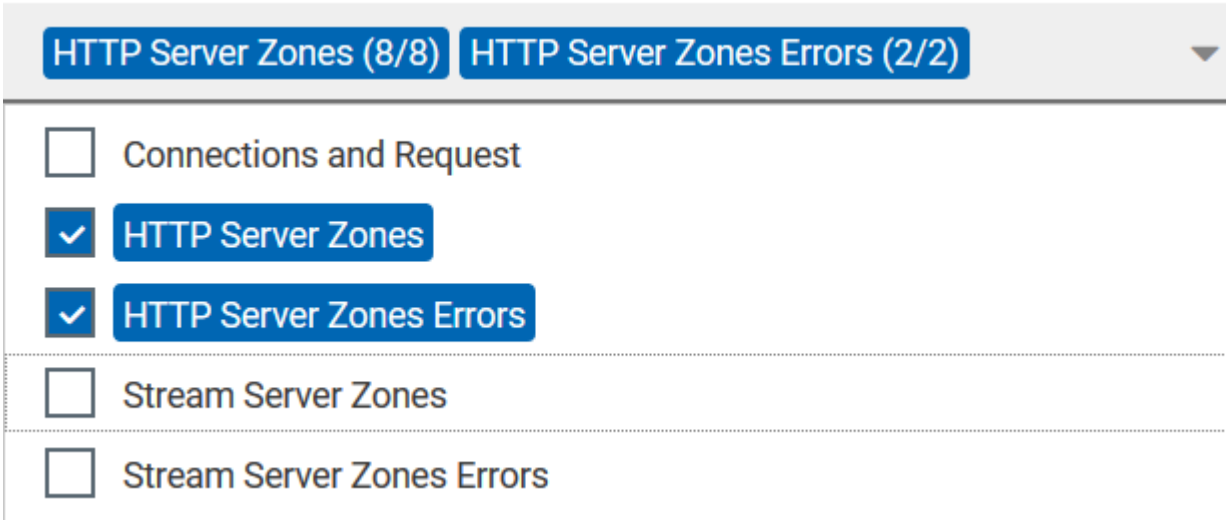
To ignore verification of the hostname in the certificate.

Require credentials

If the server requires log-in credentials, enter a **User name** and **Password**.

e. Click **Add**.
The **Choose Data Counters** dialog opens.

4. Select and save the statistics counters to monitor the source. You can select them from the list where they follow the server logical organization.
For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.



Note:

For Docker host, you can select different sets of counters.

- The first level of counters: Generic counters that are related to a Docker Image and all its running Docker Containers (in an exit, running or paused, created, restarted, removing, dead or transitive state).
- The second level of counters: Specific counters that are related to the existing Docker Containers.

Results

When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.

What to do next

You can click the links in the tables to view the performance metrics of your monitored system.

Related information

Viewing resource monitoring usage data on page 1603

Monitoring an Apache httpd server

To monitor an Apache httpd server, you must add the Apache httpd server source to Test , enter connection settings and select the counters that are used to capture the monitoring usage data.

Before you begin

You must be logged in Test and be the owner of an existing new project.

You can optionally use agents. See Resource monitoring agents on page 1581.

1. Click **Infrastructure > Resource Monitoring**.
   The resource monitoring page opens. See Resource monitoring agents on page 1581.

2. In the resource monitoring page, click **Add a Source** and select **Add an Apache httpd server...**.
3. Fill in the following connection settings in the **New server** dialog:

   Note: For hosts that are already connected to Test through an agent, you have only the **Access target from** field enabled. Move directly to step 4.

   a. In **Target host**, enter the IP address or hostname and port number of the host where the server to monitor is installed.

   b. In **Server edition**, select the appropriate server from the drop-down list.

   c. In **Path to the status page of the server**, enter the name of the page to view the status of the server.

   d. In **Security**, select the following options:

      Secured with TLS/SSL

      If the application server is secured with TLS/SSL.

      Trust self-signed certificate

      To accept the server certificate.

      Do not verify hostname

      To ignore verification of the hostname in the certificate.

      Require credentials

      If the server requires log-in credentials, enter a **User name** and **Password**.

4. Click **Add**.
   The **Choose Data Counters** dialog opens.
5. Select and save the statistics counters to monitor the source. You can select them from the list where they follow the server logical organization.
   For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.

Results

When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.

What to do next

You can click the links in the tables to view the performance metrics and resource monitoring usage data for the monitored server.

Related information

Resource monitoring on page 1563

Viewing resource monitoring usage data on page 1603

Monitoring metrics collected by a Prometheus server

You can monitor your system resources by using metrics data that are collected by a Prometheus server while executing a schedule.

Before you begin

You must have created a project in Test .

You must have installed and configured a Prometheus server.

Read the concept information about resource monitoring and monitoring Prometheus server source. See Resource monitoring on page 1563

About this task

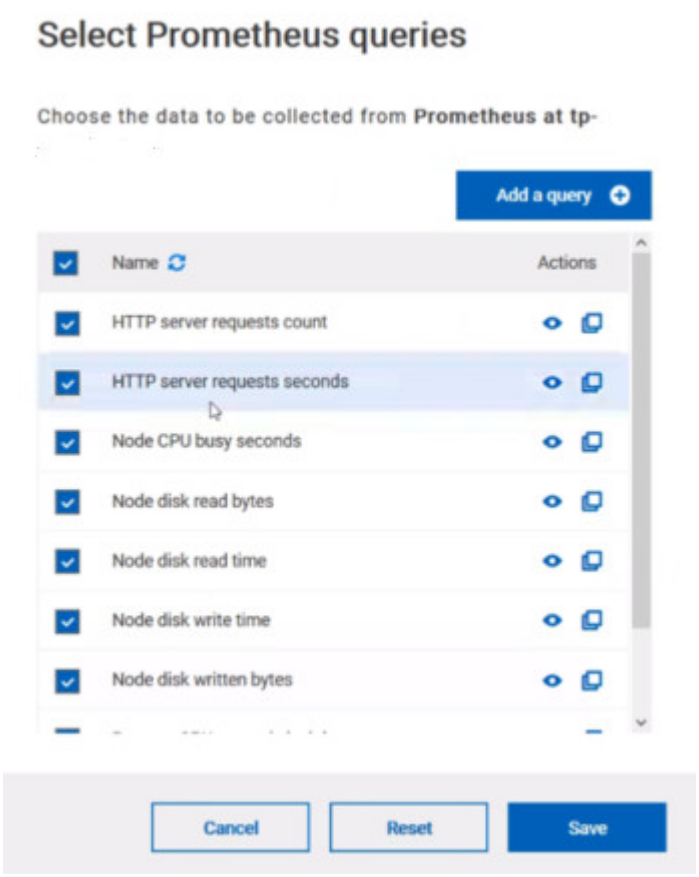
The following procedure describes how to select and create Prometheus queries in Test. You must set a connection between Test and the Prometheus server first to access to the Prometheus queries.

Enter the connection settings, then select queries.

1. Click **Add a source** and select **Add a Prometheus Server**.
2. Enter the connection settings in the **New server** dialog box,.
    a. In **Target host**, enter the IP address or host name and port number of the host where the Prometheus server is installed.
    b. Enter the **Path to the API root of Prometheus** to point to the Prometheus API where the monitoring data are saved.
    c. Enter the security parameters if required. For information about this section, see related links.
    d. Click **Add** to close the dialog.

    Result

    The Prometheus server source is created successfully in the background and a dialog opens to select the Prometheus data queries.
3. In the **Select Prometheus queries** dialog, select queries that are available in the list to collect Prometheus metrics data by clicking the checkboxes.
    You can reset the selection.



4. Follow these steps to create a query:

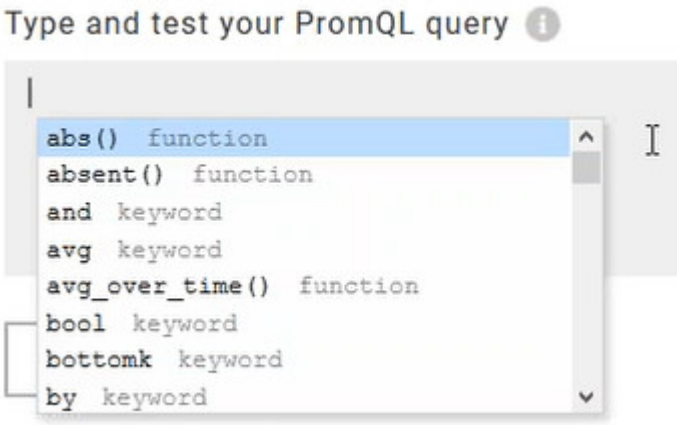
    a. Click **Add a query**.

    b. Enter a name for the new query in the **Create a Prometheus query** dialog.

    c. Enter a PromQL query.any keyword and press Ctrl+Space in the **Type and test your PromQL query.**

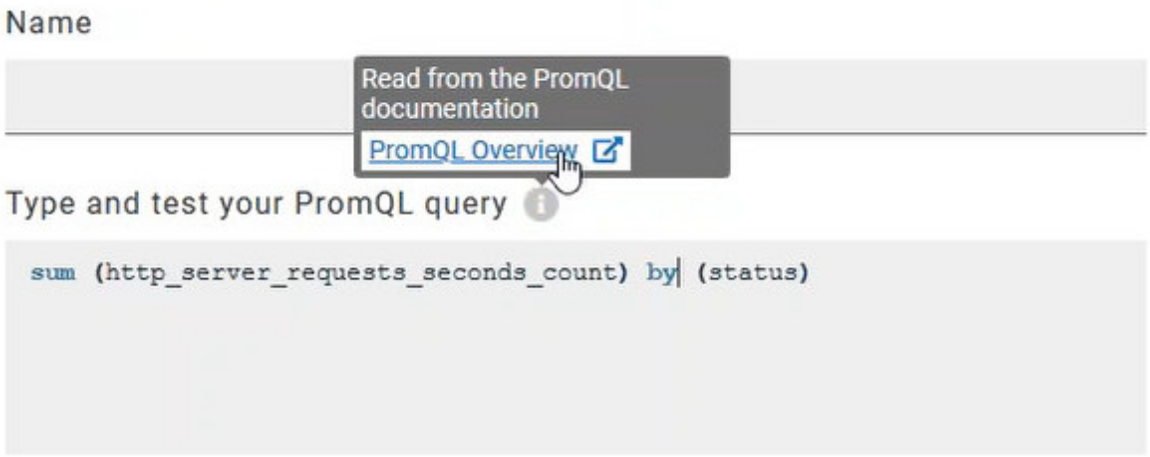
    Result

    All the possible combinations of keywords and metrics related to your searched keyword are listed.



d. Select a query in the list.

e. Click the link that points to the 'Querying Prometheus documentation' page in the information tool tip to find other examples of queries in a web page that you can use as samples.



f. Click **Test query.**
If the test is successful, a green check is displayed, otherwise you get a red check with an error message.

g. Click **Create**.

5. Proceed as follows to duplicate a query:

a. Click the **Duplicate** icon to duplicate a standard query.

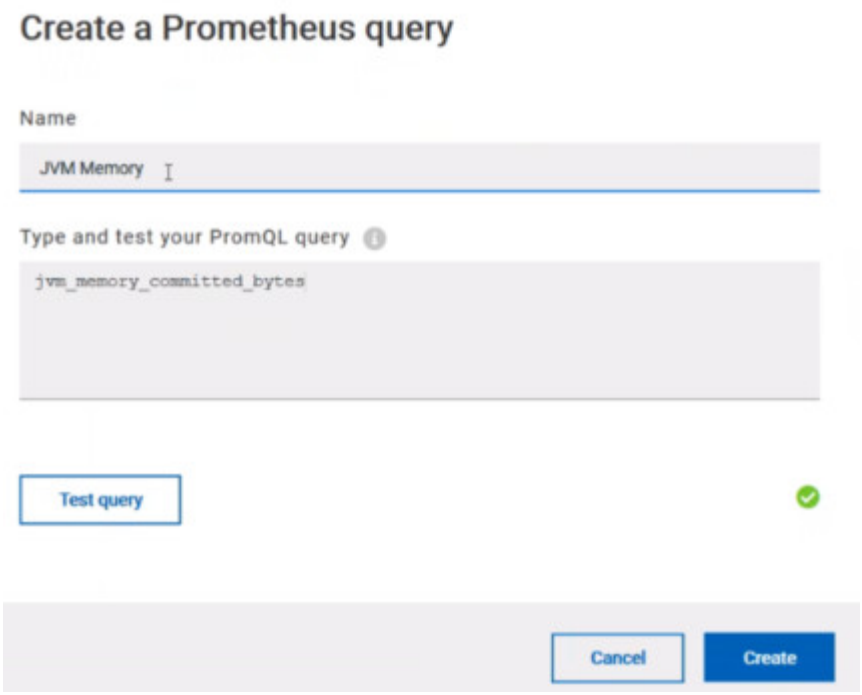
b. Rename the duplicated query in the **View the Prometheus query** dialog.
Make sure that the name you give to the query is not already used.

c. Modify the query type.

d. Test the query .

e. Click **Create**.

Note: You can click Create or Duplicate only if the test of the query is successful and the name of the query is not already in use.



6. Proceed as follows to modify a query:
   a. Click the **Modify** icon .
   b. Modify the name and type of the query.
   c. Test the query.
   d. Click **Update**.
   e. Delete duplicated or created queries.
   f. Click **Save** to close the **Select Prometheus queries** dialog.
7. Click Save and close the **Select Prometheus queries** dialog.

Results

The resource monitoring page displays the total number of sources you have added to your project and the number of sources ordered by type in cards.

What to do next

Click the links in the cards to go to the sources page and see the performance metrics for each source you added to your project. For more details, see Viewing resource monitoring usage data on page 1603

Related information

Resource monitoring on page 1563

Viewing resource monitoring usage data on page 1603

Monitoring metrics exposed by an OpenMetrics exporter

With Test , you can monitor your system resources by using metrics data that are exposed by an OpenMetrics or a Prometheus exporter through metric counters while executing a schedule.

Before you begin

You must have at least one exporter or software to expose monitoring metrics with an OpenMetrics or Prometheus format.

You must have created a project in Test.

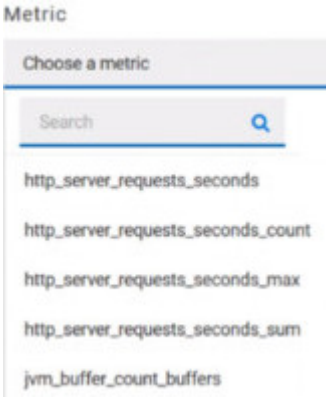
No Prometheus server installation is required, you only need to set up one or multiple OpenMetrics, or Prometheus exporters that fit your software or host target. Refer to the link at the end of this page to see the list of exporters.

About this task

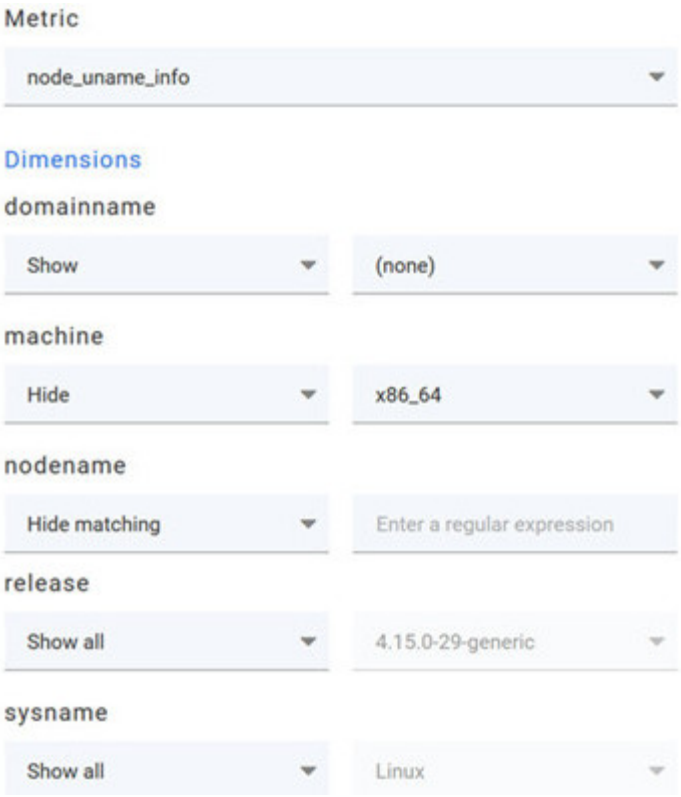
The following procedure describes how to select and create OpenMetrics counters in Test. You must set a connection between Test and the OpenMetrics exporter to select counters.

1. Click **Add a source** and select **Add an OpenMetrics exporter** .
2. In the **New OpenMetrics exporter** dialog that opens, enter the OpenMetrics exporter details to access its exposition endpoint as follows:
    a. In **Target host**, enter the IP address or host name and port number of the host where the server is installed.
    b. Enter the **Path to the exposition endpoint** to point to the application where the data performance results must be stored. As the path depends on the type of exporter used, refer to the exporter's documentation and configuration to indicate the right path.
    c. In **Security**, enter the security parameters if required.
    d. Click **Add** to close the dialog.
    Result
    The source is created successfully in the background. A dialog box opens to select the OpenMetrics counters.
3. Click the checkboxes to select counters in the **Select OpenMetrics counters** dialog.
4. Follow these steps to create a counter:
    a. Click **Add a counter**.
    b. Enter a name for the new counter, in the **OpenMetrics counter**.

c. Select metrics in the drop-down list or enter a keyword in the **Dimensions** field to enable the dynamic input help. You can also use the search field that appears in the list of metrics.

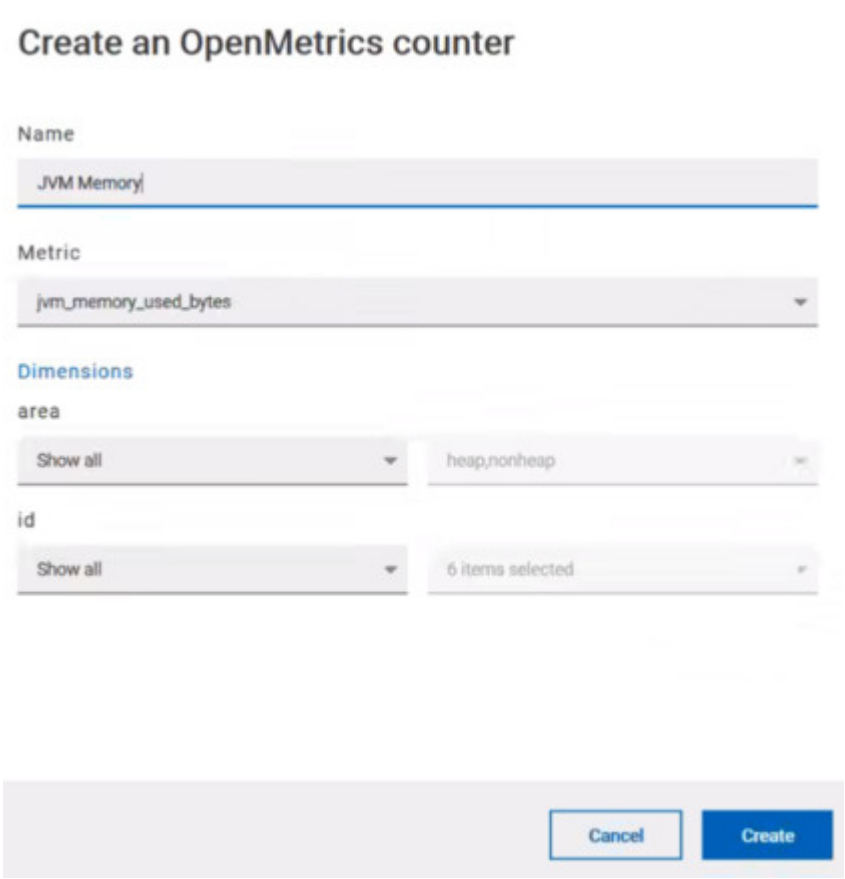


d. Select **Dimensions** from the dynamic fields when they are required.



e. Select the appropriate OpenMetrics counter that you are looking for.

f. Click **Create**.



5. Follow these steps to duplicate a counter:
    a. Select a counter in the list, and click the **Duplicate** icon .
    b. Modify the name of the counter and the Dimensions.
    c. Click **Create**.
6. Proceed as follows to modify a new counter:
    a. Click the **Modify** icon.
    b. Modify the name, Metrics, and Dimensions of the counter in the **Modify the OpenMetrics counter**.
    c. Click **Update**.

Note: You can create, duplicate, or update a counter only if the name of the counter is not already used.

7. Click **Save** and close the **Select OpenMetrics counters** dialog.

Note: You can delete counters, but only the ones that you created or duplicated.

Results

The resource monitoring page displays the total number of sources you have added to your project and the number of sources ordered by type in cards.

What to do next

Click the links in the cards to go to the sources page and see the performance metrics for each source you added to your project. For more details, see Viewing resource monitoring usage data on page 1603.

Related information

Exporters and Integrations

Resource monitoring on page 1563

Monitoring a Java Virtual Machine

To monitor a Java Virtual Machine, you must add the Java Virtual Machine source to the project in Test , and then select the counters that are used to capture the monitoring metrics.

Before you begin

You must have completed the following tasks:

- Configured and started the Java Virtual Machine from a command to connect it Test.

  See Starting a Java Virtual Machine on page 1593.
- Logged in Test. You must be the owner of an existing new project.
- You can optionally use agents. See Resource monitoring agents on page 1581.

About this task

The Java Virtual Machine source is added to the list of resource monitoring sources when the agent is installed.

1. Click **Infrastructure > Resource Monitoring**.
   The resource monitoring page opens.
2. In the resource monitoring page, click **Add a Source** and select **add a Java Virtual Machine**.
3. Fill in the following connection settings in the **New server** dialog:

   Note: For hosts that are already connected to Test through an agent, you have only the **Access target from** field enabled. Move directly to step 4.

   a. In **Target host**, enter the IP address or hostname and port number of the host where the server to monitor is installed.

   b. In **Server edition**, select the appropriate server from the drop-down list.

   c. In **Path to the status page of the server**, enter the name of the page to view the status of the server.

   d. In **Security**, select the following options:

Secured with TLS/SSL

If the application server is secured with TLS/SSL.

Trust self-signed certificate

To accept the server certificate.

Do not verify hostname

To ignore verification of the hostname in the certificate.

Require credentials

If the server requires log-in credentials, enter a **User name** and **Password**.

4. Click **Add**.
5. Select and save the statistics counters to monitor the source. You can select them from the list where they follow the server logical organization.
   For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.
   Result

   When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.

What to do next

You can click the links in the tables to view the performance metrics and usage data in a graph for the monitored Java Virtual Machine.

Related information

Resource monitoring on page 1563

Viewing resource monitoring usage data on page 1603

Monitoring a Windows Performance host

To monitor a Windows Performance host, you must add the Windows Performance host source to the project in Test , and then select the counters that are used to capture the monitoring usage data.

Before you begin

You must have completed the following tasks:

- Installed and started an agent. See Resource monitoring agents on page 1581.
- Logged in to Test. You must be the owner of an existing new project.

About this task

The Windows Performance host source is added to the list of resource monitoring sources when the agent is installed.

1. Click **Infrastructure > Resource Monitoring**.
   The resource monitoring page opens.
2. In the resource monitoring page, click **Add a Source** and select **add a Windows Performance host**.
3. Select an address to connect to the target host in **Access target from**.
4. Click **Add**.
   The **Choose Data Counters** dialog opens.
5. Select and save the statistics counters to monitor the source. You can select them from the list where they follow the server logical organization.
   For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.
   Result

   When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.

What to do next

You can click the links in the tables to view the performance metrics and usage data in a graph for the monitored Windows Performance host.

Related information

Resource monitoring on page 1563

Viewing resource monitoring usage data on page 1603

Monitoring a Linux Performance host

To monitor a Linux Performance host, you must add the Linux Performance host source to the project in Test , and then select the counters that are used to capture the monitoring usage data.

Before you begin

You must have completed the following tasks:

- Installed and started an agent. See Resource monitoring agents on page 1581.
- Logged in to Test. You must be the owner of an existing new project.

About this task

The Linux Performance host source is added to the list of resource monitoring sources when the agent is installed.

1. Click **Infrastructure > Resource Monitoring**.
   The resource monitoring page opens.
2. In the resource monitoring page, click **Add a Source** and select **add a Linux Performance host**.
3. Select an address to connect to the target host in **Access target from**.
4. Click **Add**.
   The **Choose Data Counters** dialog opens.
5. Select and save the statistics counters to monitor the source. You can select them from the list where they follow the server logical organization.
   For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.
   Result

When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.

What to do next

You can click the links in the tables to view the performance metrics and usage data in a graph for the monitored Linux Performance host.

Related information

Resource monitoring on page 1563

Viewing resource monitoring usage data on page 1603

Resource monitoring agents

You can connect agents to Test from other hosts to monitor a larger set of data collectors, computers, servers, or systems, or if a server to monitor requires authorization access. To monitor these remote servers, you must configure and run the agents so that a connection is set between Test resource monitoring service and the agents. You can configure the resource monitoring agents in a team space or in a project.

Requirements:

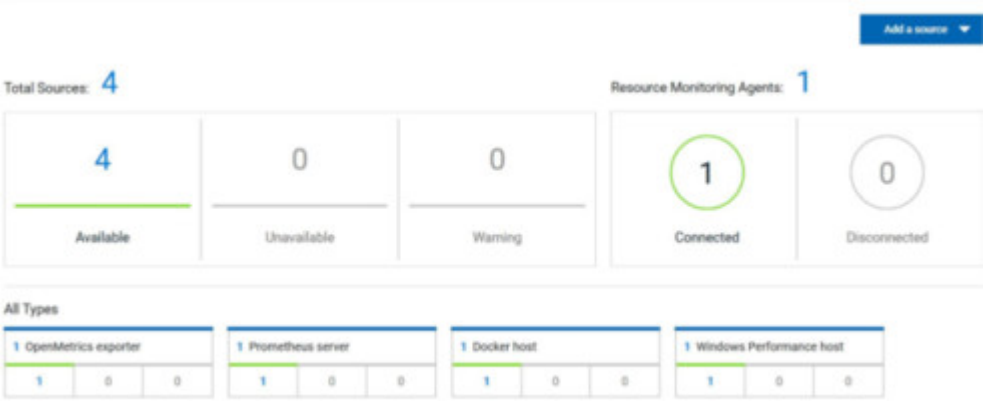
- To connect resource monitoring agents to Test in a team space, you must be a *Team Space Owner*. Any other team space *Member* (not owner) can only view the resource monitoring agents.
- To connect resource monitoring agents to Test in a project, you must be a project *Owner*. Any project *Member* (not the owner) can only view the resource monitoring agents.
- Agent-based monitoring is mandatory to collect data from a Windows or Linux Performance host.

For testing, you might need many hosts. For example, you might have one host with the application server, another host with the database server, and some hosts to apply the user load. Due to network or firewall issues, sometimes, it becomes difficult for multiple hosts to connect to each other. Resource monitoring agents are installed on the target hosts so that they can establish a connection with Test to gather resource statistics of the target host.

The agents always try to connect with the server through the HTTPS protocol. You must install the agent and start it. From the resource monitoring page in Test, you can copy the command lines to download files and run the agents. When you stop the agent, the monitoring sources that you have already added persist but the live data will not be available.

You can configure your resource monitoring agent as a service so that the agent automatically runs when the host is restarted. For details, see the documentation pages about starting resource monitoring agents on Windows or Linux.

When the agent is connected to Test, it is added to the main page of the resource monitoring service. Then, you can select the agent to connect to a server from **Add a Source**.



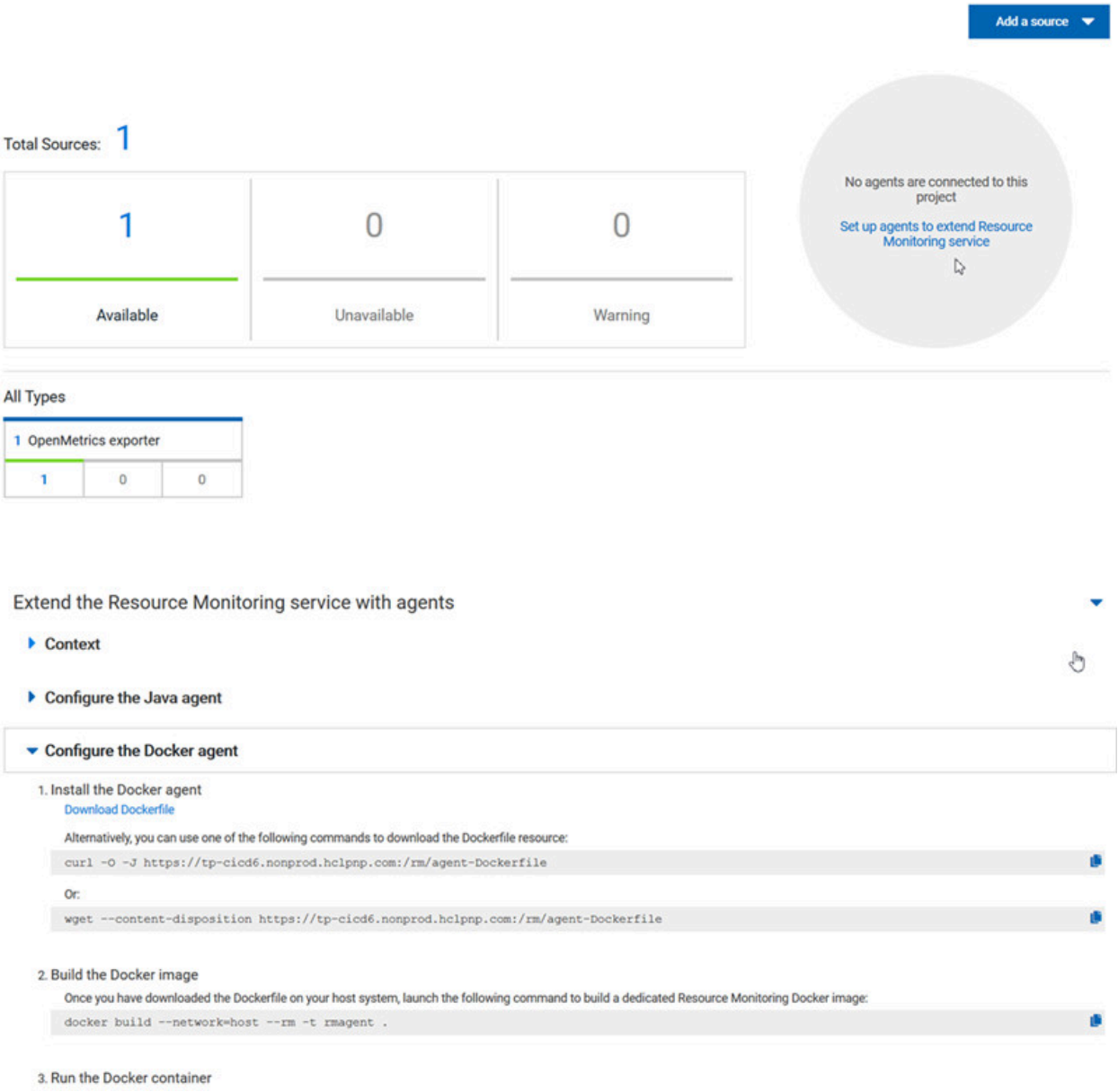
Installing and running a Docker agent on a remote host

You must install the resource monitoring Docker agents on the target host for which you want to monitor the resources and collect the metrics. You need to run an agent to establish a connection with Test .

Before you begin

- The resource monitoring service does not require access to the agent host but the agent must have reached the service host over HTTPS.
- The Docker agent must have been launched in a Docker container and requires Docker 19.03.
- Read and been familiar with the requirements to install and configure the resource monitoring agents on Test. See Resource monitoring agents on page 1581.

1. Click **Set up Agents to extend Resource Monitoring service** link in Test to access the agents page where you can find the instructions and commands that are to be used to install and run the agents.



2. Expand **Configure the Docker agent** and click **Download Dockerfile** to download a Docker agent.
   You can also use the *curl* or *wget* commands to download the agent without accessing the resource monitoring web UI.

   For more facility, use the code snippets to copy and fill in the commands with the valid offline token and the Docker file name.

   For Windows:

   ```
   curl -O -J https://hostName/rm/agent-Dockerfile
   ```

   For Linux:

```
wget --content-disposition hostName/rm/agent-Dockerfile
```

3. Copy the build command that is under **Build the Docker image**, paste it in your console and run it to build a resource monitoring Docker image.
   Before you start the a Docker agent, you must build a Docker image.

   ```
   docker build --network=host --rm -t rmagent
   ```

4. Follow these steps to run the agent:

   a. Click to fill in the appropriate command for Windows or Linux with the valid offline token.

   b. Click to copy the command to clipboard and paste the command in the console.

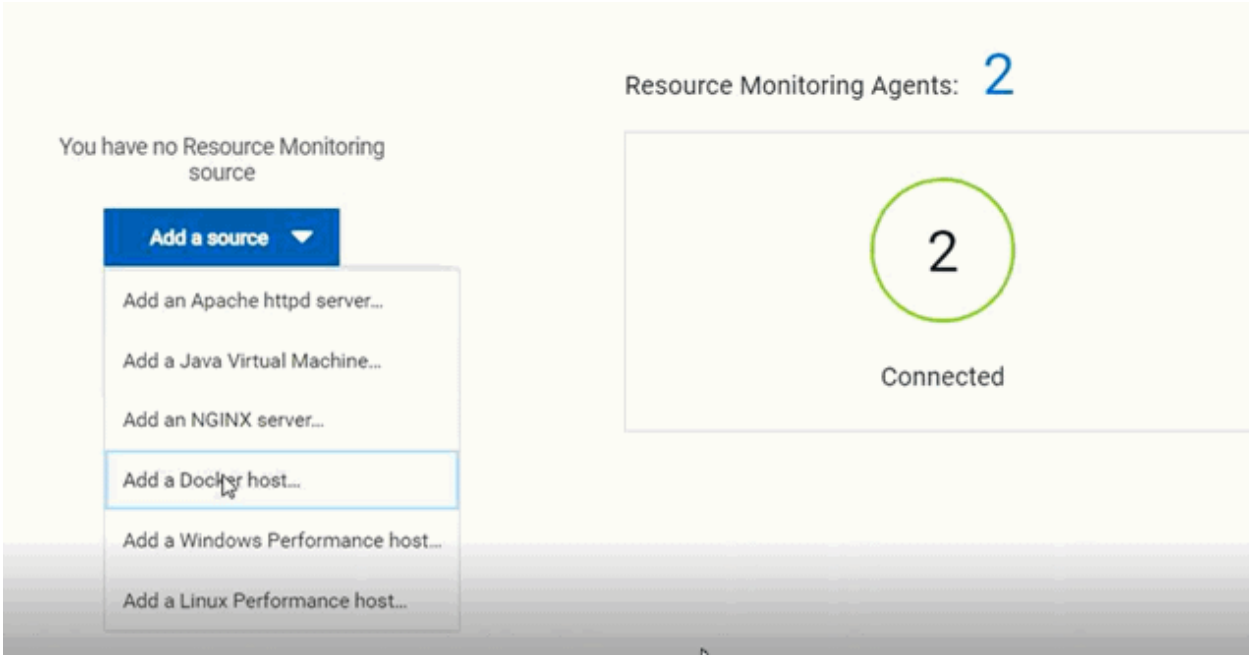
   - Command on Windows:

   ```
   docker run -it --rm --privileged -e PROJECT_ID=230200 -e HCL_ONETEST_OFFLINE_TOKEN=(Enter your
    offline token here) --network host -v //var/run/docker.sock:/var/run/docker.sock rmagent
   ```

   - Command on Linux:

   ```
   docker run -it --rm --privileged -e PROJECT_ID=230200 -e HCL_ONETEST_OFFLINE_TOKEN=(Enter your
    offline token here) --network host -v /var/run/docker.sock:/var/run/docker.sock -v /etc:/etc
    -v /proc:/proc rmagent
   ```

   c. Paste the command in the console and run the command.



Results

When the agent is started, the agent is connected to Test. The agent is displayed in the list of connected resource monitoring agents with the hostname and status of the agent. The agent is also added to the main page of the resource monitoring service in Test.

What to do next

You can add a new resource monitoring source. It can be collected from a service or from the named agents, depending on the capabilities supported by the environment.

Related information

Resource monitoring capabilities on page 1564

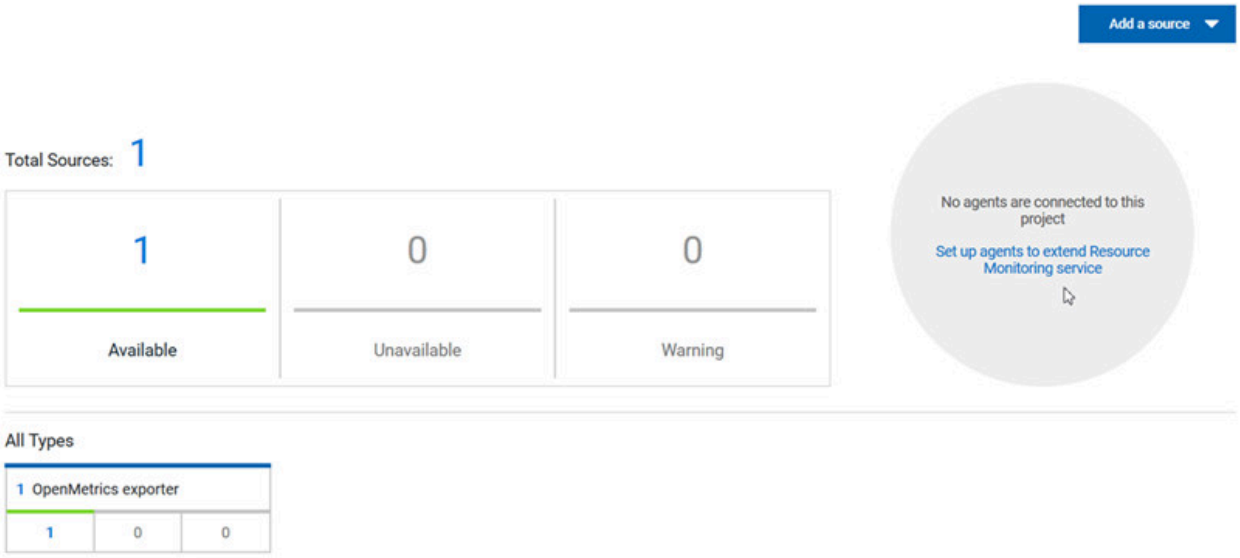
Installing and running a Java agent on a remote host

You must install the resource monitoring java agent on the target host for which you want to monitor the resources and collect the metrics of Test . You need to run the agent to establish a connection with Test.

Before you begin

- The resource monitoring service does not require access to the agent host but the agent must have reached the service host over HTTPS.
- The Java agent must have been launched from a jar file and requires a Java 17 virtual machine.
- Read and been familiar with the requirements to install and configure the resource monitoring agents on Test. See Resource monitoring agents on page 1581.

1. Click **Set up Agents to extend Resource Monitoring service** link in Test to access the agents page where you can find the instructions and commands that you can use to install and run the agents.



2. Expand **Configure the Java agent** and click **Download jar file** to download the Java agent.
   You can also use the *curl* and *wget* commands to download the agent without accessing the resource monitoring web UI. For more facility, use the code snippets to copy and fill in the commands with the valid offline token and the jar file name.

With curl:

```
curl -O -J https://hostName/rm/Agent-jar
```

With wget:

```
wget --content-disposition https://hostName/rm/Agent-jar
```

3. Configure the command as follows to run your agent:
    a. Copy the appropriate command for Windows or Linux with the code snippets.
    b. Enter the path to the directory that contains the agent .jar file, and paste the command.

Example

To connect resource monitoring agents to Test in a team space use the following commands:

On Windows:

```
set HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here)
java -jar (Enter the name of the downloaded jar file here) --ServiceUrl=https://hostName/rm
 --spaceId=<team-space_id>
```

On Linux:

```
sudo HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here) java -jar (Enter the name of the
 downloaded jar file here) --ServiceUrl=portNumber/rm --spaceId=<team-space_id>
```

Example

To connect resource monitoring agents to Test in a project use the following commands:

On Windows:

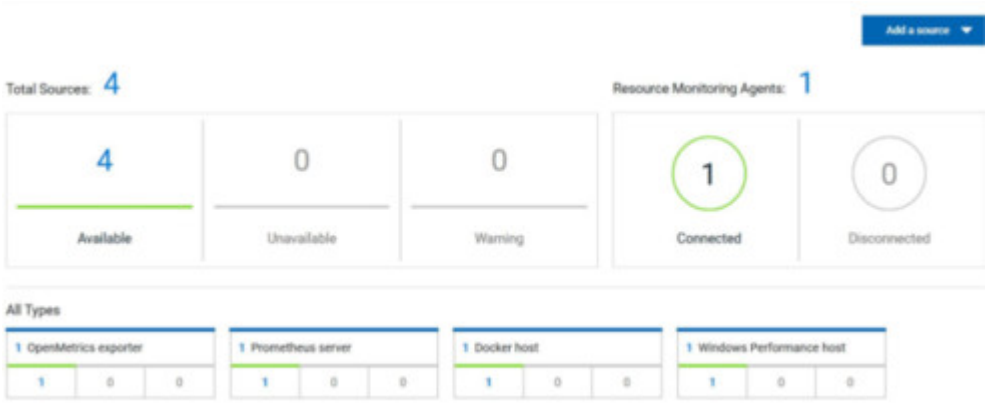
```
set HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here)
java -jar (Enter the name of the downloaded jar file here) --ServiceUrl=https://hostName/rm
 --projectId=<project_id>
```

On Linux:

```
sudo HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here) java -jar (Enter the name of the
 downloaded jar file here) --ServiceUrl=portNumber/rm --projectId=<project_id>
```

Results

When the agent is started, the agent is connected to Test. The agent is displayed in the list of connected resource monitoring agents with the hostname and status of the agent. The agent is also added to the main page of the resource monitoring service in Test.



What to do next

You can use another method to start an agent. See Starting a resource monitoring Java agent as a service on Windows on page 1587 or Starting a resource monitoring Java agent as a service on Linux on page 1590. You can then add a new resource monitoring source. Resource monitoring metrics can be collected from a service or from the named agents, depending on the capabilities supported by the environment.

Related information

Resource monitoring capabilities on page 1564

Starting a resource monitoring Java agent as a service on Windows

To ensure that the resource monitoring Java agent starts by itself when the host is running, you can set up the environment in such a way that the resource monitoring agent can be started as a service.

Before you begin

You must have completed the following tasks:

- Installed Java 17 on the host.
- Added Test to the PATH environment variable.
- Created an offline token to connect the agent securely with appropriate permissions.

Note: You can create an offline token from the User menu of the resource monitoring page or you can re-use your active offline token. The token expires if it is not used for a month.

To allow remote access to the performance data on Windows 10, you must have considered the following information:

- The username and password are the same on the local and remote servers. Otherwise you must have provided remote server credentials.
- The remote user must be a member of the Performance Monitor Users group (start `lusrmgr.msc` and add the user to this group).
- The Remote Registry service must be running on the remote host (start `services.msc`) and must have verified the Remote Registry status).
- File and printer sharing must be enabled on the Network Interface of the remote host that is implied in the communication with the local host.
- You must have activated the following Windows Firewall rules so that the remote firewall does not block the access:
  - File and Printer Sharing (NB-Name-In)
  - File and Printer Sharing (NB-Session-In)
  - File and Printer Sharing (*)

1. Download the latest release of *winsw*.
   You can choose to download `WinSW.Net2.exe` or `WinSW.Net4.exe` depending on the version of .Net framework that you already have on the host Windows.
2. Create a folder on your local hard drive like `RMAgent-winservice`.
3. Copy the downloaded executable to this new folder and rename the file to `RMAgent-winservice.exe`.
4. Create a new text file in the same folder and name it `RMAgent-winservice.xml`.
5. Copy and adapt the following content to this new RMAgent-winservice.xml file to set up the offline token:

```
<service>
      <id>RMAgent-winservice</id>
      <name>Resource monitoring agent</name>
      <description>This service runs the resource monitoring agent.</description>
      <executable>java</executable>
      <env name="HCL_ONETEST_OFFLINE_TOKEN" value="(Enter your offline token here"/>

      <arguments>-jar %BASE%\RMAgent.jar --serviceUrl=https://<service-host>/rm --projectId=<projet-id>
 --autoUpgradeDownloadThen=execute:cmd,/c,start,%BASE%\auto-upgrade.bat</arguments>
      <onfailure action="restart" delay="10 sec"/>
      <logmode>rotate</logmode>
</service>
```

Notes:

- Replace *<service-host>* by the hostname of the host that runs Test.
- Replace *<project-id>* which is the number that you find after /projects/ in the browser's URL when you browse to this project.

6. Create a new text file in the same folder and name it `auto-upgrade.bat`.
7. Copy the following content to this new file to upgrade the agent `.jar` file automatically:

```
@echo off
for /f "tokens=*" %%a in ('dir /b /od %BASE%\com.hcl.test.rm.agent-*.jar') do set newest=%%a
%BASE%\RMAgent-winservice.exe stop
del %BASE%\RMAgent.jar
mklink %BASE%\RMAgent.jar %BASE%\%newest%
%BASE%\RMAgent-winservice.exe start
```

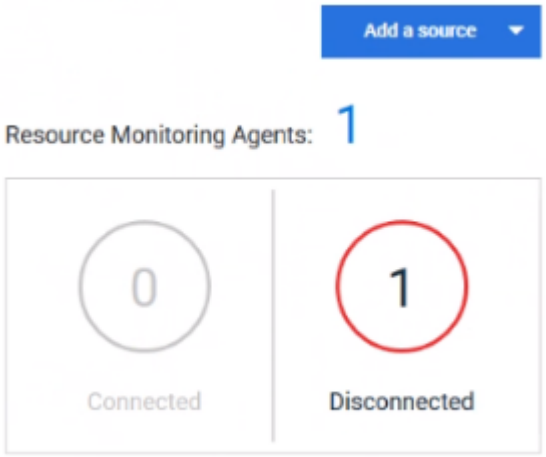
8. Start the command prompt as an administrator and change the directory to the newly created directory: `RMAgent-winservice`.
9. Download the agent .jar file that is available from the agents page in the resource monitoring service, under the **Extend the Resource Monitoring service with agents** section and save it to the same directory.
10. Enter the following command in the command prompt to create a symbolic link named `RMAgent.jar` to the agent jar file:

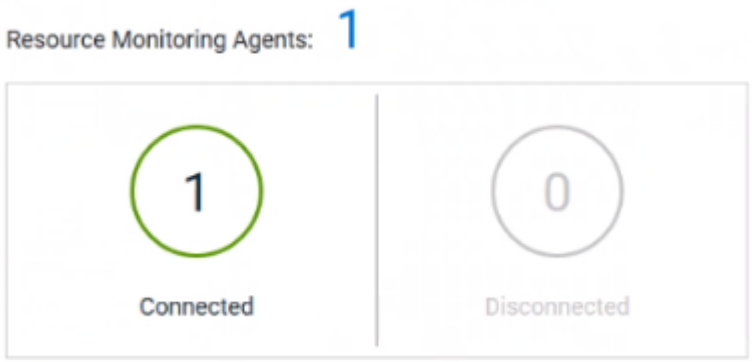
    ```
    mklink RMAgent.jar com.hcl.test.rm.agent-<version-and-datetime>.jar
    ```

11. Enter the following commands:

    ```
    RMAgent-winservice install
    ```

    ```
    RMAgent-winservice start
    ```





Notes:

In the Services Windows application, you can check whether the service is up and running. You can see in the resource monitoring page whether the resource monitoring agent is disconnected or connected.

The resource monitoring agents write the logs to the same folder in files named `RMAgent-winservice.out.log`, `RMAgent-winservice.err.log` and `RMAgent-winservice.wrapper.log`. If the Windows service for the agent is not started or if it is not connected to the resource monitoring service, verify the log files.

Results

The Java agent starts as a service on Windows. When the agent is connected to Test, it is automatically added to the list of resource monitoring sources in Test.

What to do next

You can monitor the Windows agent by selecting counters. See Monitoring a Windows Performance host on page 1579.

Related information

Other commands

winsw file

Resource monitoring capabilities on page 1564

Starting a resource monitoring Java agent as a service on Linux

To ensure that the resource monitoring Java agent starts by itself when the host is restarted, you can set up the environment in such a way that the resource monitoring agent can be started as a service.

Before you begin

You must have completed the following tasks:

- Installed Java 17 on the host.
- Added Test to the PATH environment variable.
- Created an offline token to connect the agent securely with appropriate permissions.

Note: You can create an offline token from the User menu of the resource monitoring page or you can re-use your active offline token. The token expires if it is not used for a month.

About this task

This topic relies on systemd services that are the default on most modern Linux distributions. Other ways mi require adaptations of the instructions but the provided script will be a good basis in most cases.

1. Create a folder on your local hard drive like `/opt/RMAgent-linuxservice`.
2. Download the agent .jar file that is available from the Agents page in the resource monitoring service, below the **Extend the Resource Monitoring service with agents** section and save it to the same directory.
3. Create a new file `/etc/systemd/system/RMAgent-linuxservice.service`.

4. Add the following content to this new file:

```
[Unit]
Description = Resource monitoring agent
After = network.target

[Service]
Type = forking
ExecStart = /opt/RMAgent-linuxservice/RMAgent-linuxservice.sh start
ExecStop = /opt/RMAgent-linuxservice/RMAgent-linuxservice.sh stop
Restart = on-failure
RestartSec = 10

[Install]
WantedBy = multi-user.target
```

5. Create another file `/opt/RMAgent-linuxservice/RMAgent-linuxservice.sh`.

```
#!/bin/sh

#!/bin/sh

# Update the 3 following variables with the Server's hostname, project id and offline token:
SERVICE_URL=https://<hostname>/rm
PROJECT_ID=<project-id>
export HCL_ONETEST_OFFLINE_TOKEN=<offline-token>

ARGS="--serviceUrl=$SERVICE_URL --projectId=$PROJECT_ID --autoUpgradeDownloadThen=exitFailure"
SCRIPT=$(readlink -f "$0")
RMAGENT_HOME=$(dirname "$SCRIPT")

# Ensure we're using the latest downloaded jar file
PATH_TO_JAR=`ls -t $RMAGENT_HOME/com.hcl.test.rm.agent-*.jar | head -1`
if [ -z "$PATH_TO_JAR" ]
then
  cd $RMAGENT_HOME && { curl -k -O -J $SERVICE_URL/agent-jar; cd -; }
  PATH_TO_JAR=`ls -t $RMAGENT_HOME/com.hcl.test.rm.agent-*.jar | head -1`
  if [ -z "$PATH_TO_JAR" ]
  then
    echo "Start the server at $SERVICE_URL to allow download of the latest agent jar file"
    echo "Exiting..."
    exit 1
  fi
fi
SERVICE_NAME="Resource monitoring agent"
#Pid file will reside in this script’s folder
PATH_TO_PID=$RMAGENT_HOME/RMAgent-pid
#Log file will reside in this script’s folder
PATH_TO_LOG=$RMAGENT_HOME/RMAgent.log

case $1 in
    start)
        echo "Starting $SERVICE_NAME ..."
        if [ ! -f $PATH_TO_PID ]; then
            nohup java -jar $PATH_TO_JAR $ARGS >> $PATH_TO_LOG 2>&1 &
                        echo $! > $PATH_TO_PID
            echo "$SERVICE_NAME started ..."
```

```
        else
            echo "$SERVICE_NAME is already running ..."
        fi
    ;;
    stop)
        if [ -f $PATH_TO_PID ]; then
            PID=$(cat $PATH_TO_PID);
            echo "$SERVICE_NAME stopping ..."
            kill $PID;
            echo "$SERVICE_NAME stopped ..."
            rm $PATH_TO_PID
        else
            echo "$SERVICE_NAME is not running ..."
        fi
    ;;
    restart)
        if [ -f $PATH_TO_PID ]; then
            PID=$(cat $PATH_TO_PID);
            echo "$SERVICE_NAME stopping ...";
            kill $PID;
            echo "$SERVICE_NAME stopped ...";
            rm $PATH_TO_PID
            echo "$SERVICE_NAME starting ..."
            nohup java -jar $PATH_TO_JAR $ARGS >> $PATH_TO_LOG 2>&1 &
                        echo $! > $PATH_TO_PID
            echo "$SERVICE_NAME started ..."
        else
            echo "$SERVICE_NAME is not running ..."
        fi
    ;;
esac
```

Notes:

- Replace *<service-hostname>* by the hostname of the host that runs Test.
- Replace *<project-id>* which is the number you'll find after `/projects/` in the browser's URL when browsing to this project.

Results

The resource monitoring Java agent starts automatically when the host restarts. The Linux agent is added to the main page of the resource monitoring service in Test.

What to do next

You can monitor the Linux Performance host source by adding counters. See Monitoring a Linux Performance host on page 1580.

Related information

systemd services

Resource monitoring capabilities on page 1564

Starting a Java Virtual Machine

To monitor the sources from a Java Virtual Machine (JVM), you must first start the JVM. You must enter the IP address of the JVM, the IP port, and the security parameters in the command before running the Java Virtual Machine.

1. Enter The IP address of the JVM (local or remote host) and the IP port in the command that is used to run the Java Virtual Machine.
2. **Optional:** You can use the authentication security data to start the virtual machine. In this case, you must enter the name of the password file. You can also enter the name of an access file that might be needed if user privileges are required.

Example

Parameters used to launch a JVM without security:

```
java
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-jar MyapplicationFile.jar
```

Parameters used to launch a JVM with authentication security:

```
java
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=true
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.password.file=jmxremote.password
-Dcom.sun.management.jmxremote.access.file=jmxremote.access (this command is optional)
-jar MyapplicationFile.jar
```

Related information

Resource monitoring agents on page 1581

Monitoring a Java Virtual Machine on page 1577

Managing sources

After you add and connect monitoring sources to your project, you might want to modify the sources in you project and the configuration of the sources.

Before you begin

You must be a project owner or team space owner to add or delete sources.

About this task

In the page where you can view the resource monitoring usage data and metrics, you can also manage your sources. You can modify or delete a source, add or change statistic counters, add labels to identify the sources and use them when you execute a test to collect resource monitoring results.

1. Click **Infrastructure > Resource Monitoring** in the navigation pane.
   If you have sources and agents in your project, you can see the cards with the number of available sources and agents in your project.
2. Click a link in the card of available sources to open a list of sources.
3. Perform any of the following actions for a monitoring source:
   - Click the **Delete** icon to delete the monitoring source and click **Delete** to confirm the deletion in the dialog box.

     The source is removed from the list of sources.
   - Click the **Edit** icon to change the target host settings.

     You can change the target host address, the port, the path to the server status page and the security parameters.
   - Click the **Selection** icon to select additional counters or change the selected counters for other ones. The changes are reflected in the resource monitoring usage data graph.
   - Click to add a label to identify the source.
   - Click **Add a source** to add a source to your project from the list of sources.

Related information

Resource monitoring capabilities on page 1564

Viewing resource monitoring usage data on page 1603

Creating labels

After you connect a host or service to a project, you can add monitoring labels to identify the monitoring sources that are used when you execute a schedule from Test . You can also use labels in any test to collect resource monitoring usage data during a test run for a defined period of time.

Before you begin

You must have connected sources to your project in Test. See Resource monitoring capabilities on page 1564.

About this task

You must be a project owner to add labels to the resource monitoring sources in a project and you must be a team space owner to add labels in a team space.

1. Click a source that is available in your project, either in the **Total Sources** cards or in the **All Types** cards from the resource monitoring page.
2. Click under the **Labels** column of a resource monitoring source to add a label.
3. Enter a name for the label and press the Enter key.

Note: You can add one label or multiple labels for the same resource monitoring source. A label can contain a maximum of 200 characters.

Tip: You can create the same label for different sources that belong to the same logical group to monitor the whole group with only one label.

Results

The host source is identified by a label.

What to do next

You can use the created labels in tests to collect resource monitoring results or persistent usage data. See Controlling resource monitoring sources in a schedule on page 1595 and Collecting and viewing resource monitoring usage data on page 1601.

Controlling resource monitoring sources in a schedule

In Test , you can control the resource monitoring sources that are collected in a performance schedule and override them with the labeled resource monitoring sources that are available in your current Server project.

Before you begin

- You must have performed the following tasks in HCL DevOps Test Performance (Test Performance):
  - Created a performance schedule of **Rate** or **VU** type.
  - Added resource monitoring sources, or the labels of resource monitoring sources to a schedule. See Adding resource monitoring sources to a performance schedule by using labels.
  - Enabled Test as resource monitoring service only if you want to replace resource monitoring labels set in Test Performance with the labels that you created in Test.
- You must have created a project and added resource monitoring sources to your project in Test. See Resource monitoring capabilities on page 1564.

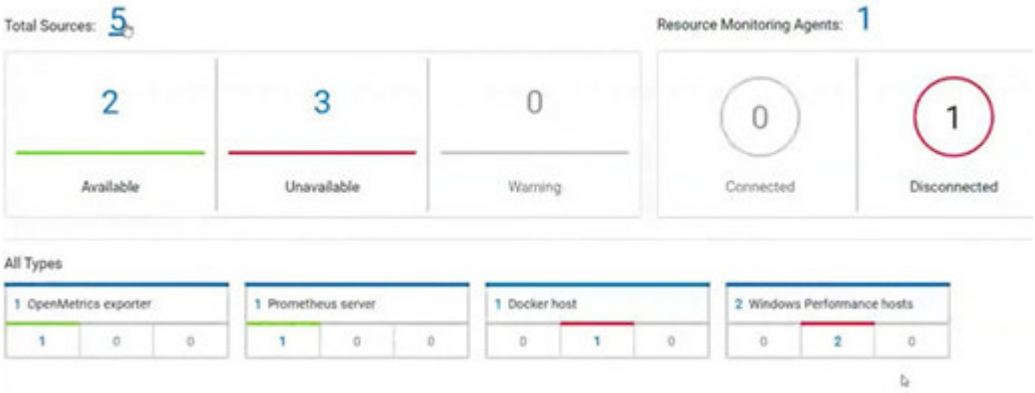
About this task

This procedure describes how to create resource monitoring labels in Test and how to use them to override the source or label settings related to resource monitoring in a performance schedule. The procedure applies to the following use cases:

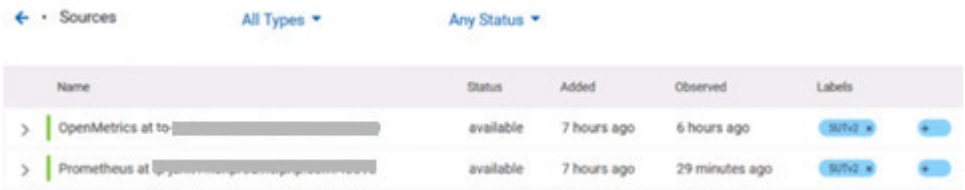
- To replace the resource monitoring labels that were initially set in a schedule from Test Performance with the labels that you create in Test.
- To enable resource monitoring sources that were not enabled from Test Performance. In that case, you can enable resource monitoring sources from the **Execution** dialog in Test.
- To ignore resource monitoring sources that were set in the schedule and to change them for labels that match sources that are available in the Test project.

Notes: You must be a project owner to add labels to the resource monitoring sources in a project and you must be a team space owner to add labels in a team space. You must have a project owner or a tester role access to add labels to a performance schedule in a project.

1. Click a source that is available in your project, either in the **Total Sources** cards or in the **All Types** cards from the resource monitoring page.



2. Click + under the **Labels** column of a resource monitoring source to add a label.



3. Enter a name for the label and press the Enter key.

   Example: Enter 'SUTv2' in three different resource monitoring sources.

Note: You can add one label or multiple labels for the same resource monitoring source. A label can contain a maximum of 200 characters.

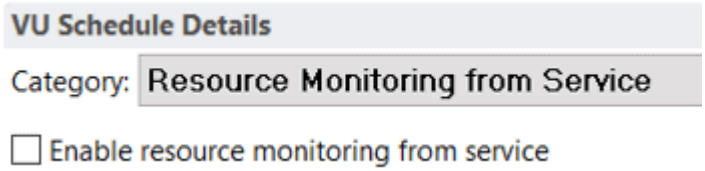
Tip: You can create the same label for different sources that belong to the same logical group to monitor the whole group with only one label.

4. Click **Execution** in the navigation pane and click on a performance schedule in the list of tests to execute a schedule.
5. Click the **Resource Monitoring** tab in the **Execute Test Asset** dialog.

Note: There are different use case scenarios:

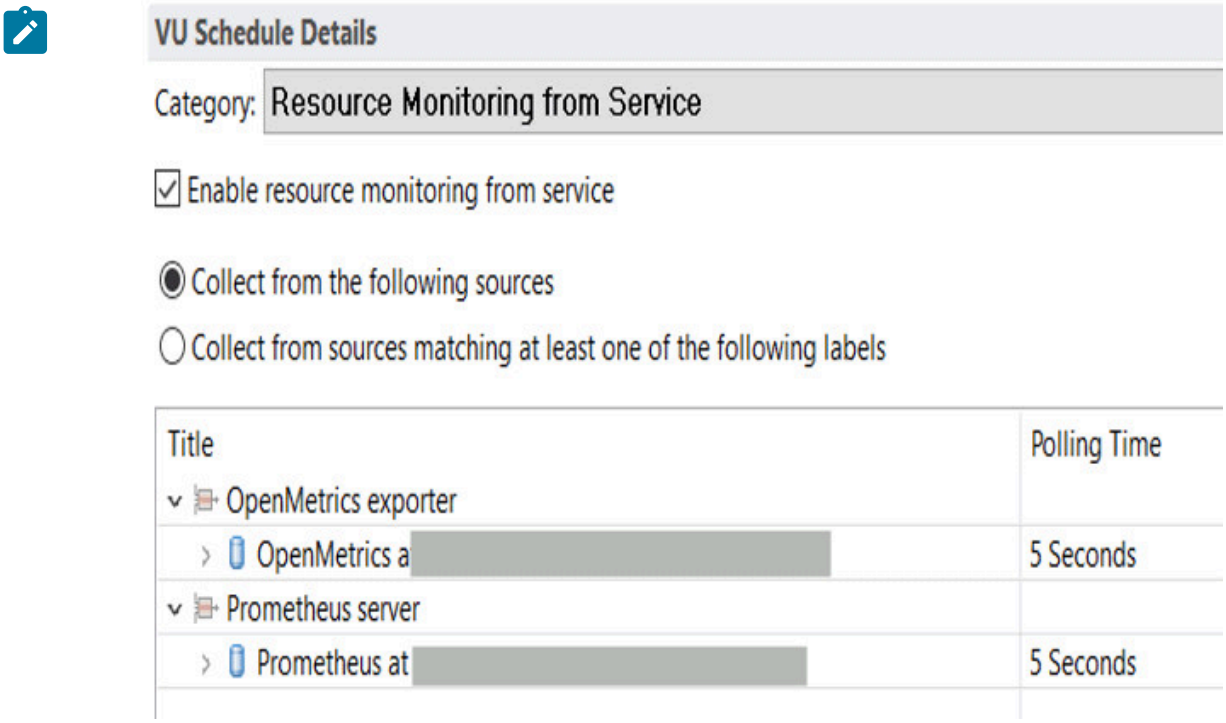
- If the **Resource Monitoring from Service** option was not enabled in Test Performance (see Figure 1), no source is available in Test project.

  Figure 1:

  A message indicates in the **Resource Monitoring** tab that Resource Monitoring was not enabled. You can run the schedule with the resource monitoring sources that are created and labeled in Test (steps 1 to 6 in this procedure) by selecting labels that are in your Test project (step 6).
- If the **Resource Monitoring from Service** option was enabled in Test Performance, and data is collected from the sources you added to the performance schedule in Test Performance (see Figure 2), the **Resource Monitoring** tab displays the list of these sources that are available in your Test project.

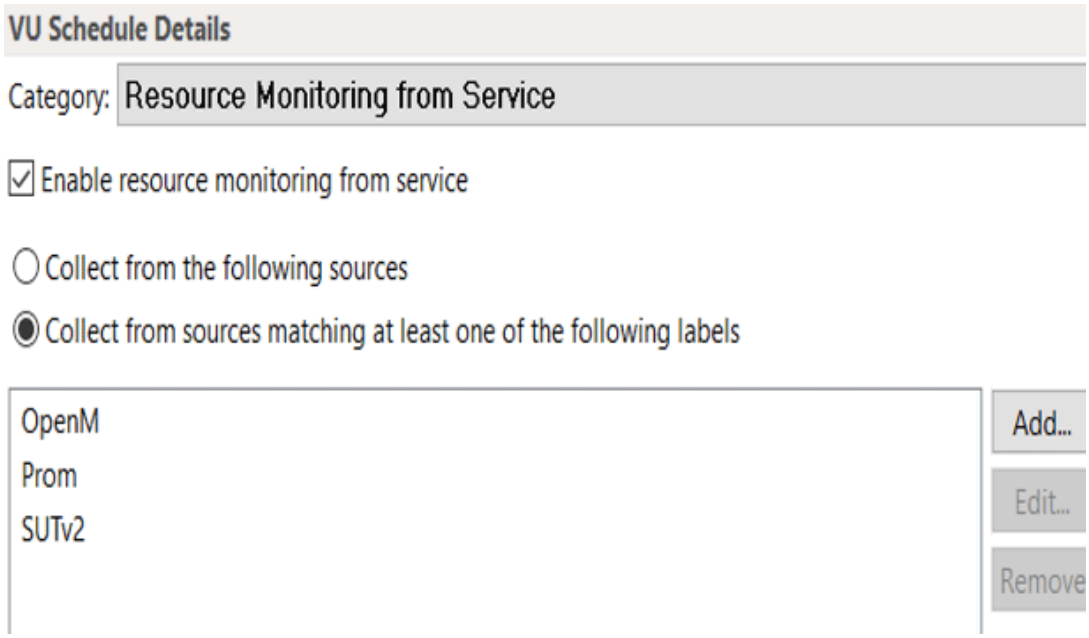
  Figure 2:



To override the initial sources with the sources that you created in Test, select the labels that are in your Test project and that match the Server sources (step 6 in this procedure).

- If you added sources to your performance schedule by using labels in Test Performance (see Figure 3), the **Resource Monitoring** tab displays the list of labels that are available in your Test project.

Figure 3:



To select resource monitoring labels from your schedule, follow step 6 in this procedure.

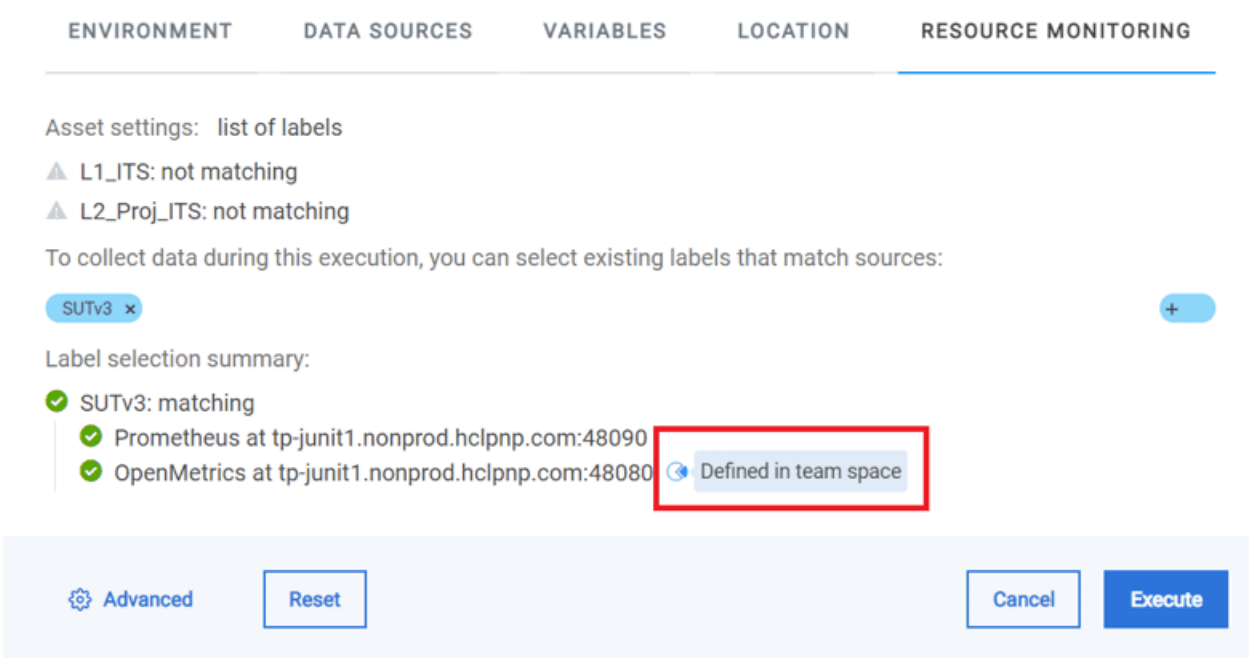
In the list of sources or labels that is displayed in the **Resource Monitoring** tab:

- Green icons identify either labels that match the sources or sources that are found in the current project.
- Red icons indicate that the sources are not running,
- Yellow icons identify either labels that do not match the sources, or sources that are not found in the current project.

  Data is collected for the sources that match labels only.

6. Click the + , and press the `Ctrl` + `Space` keys, or enter the initial letter of a label to select a label in the list.

   Example: Select **SUTv3**.

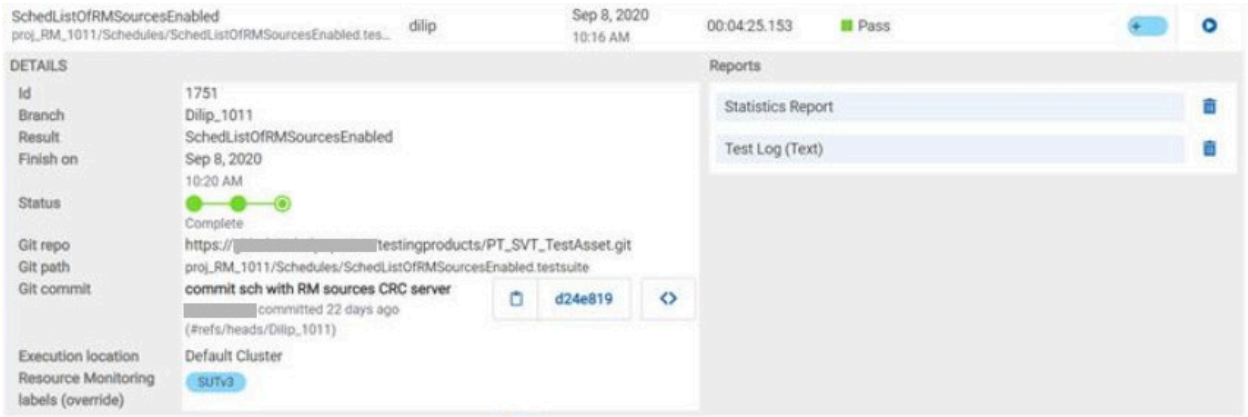


   You can see all the labels that you added to the resource monitoring sources in Test.

Note: You can select labels matching resource monitoring sources that are created in a team space in addition to the ones that are created in a project.

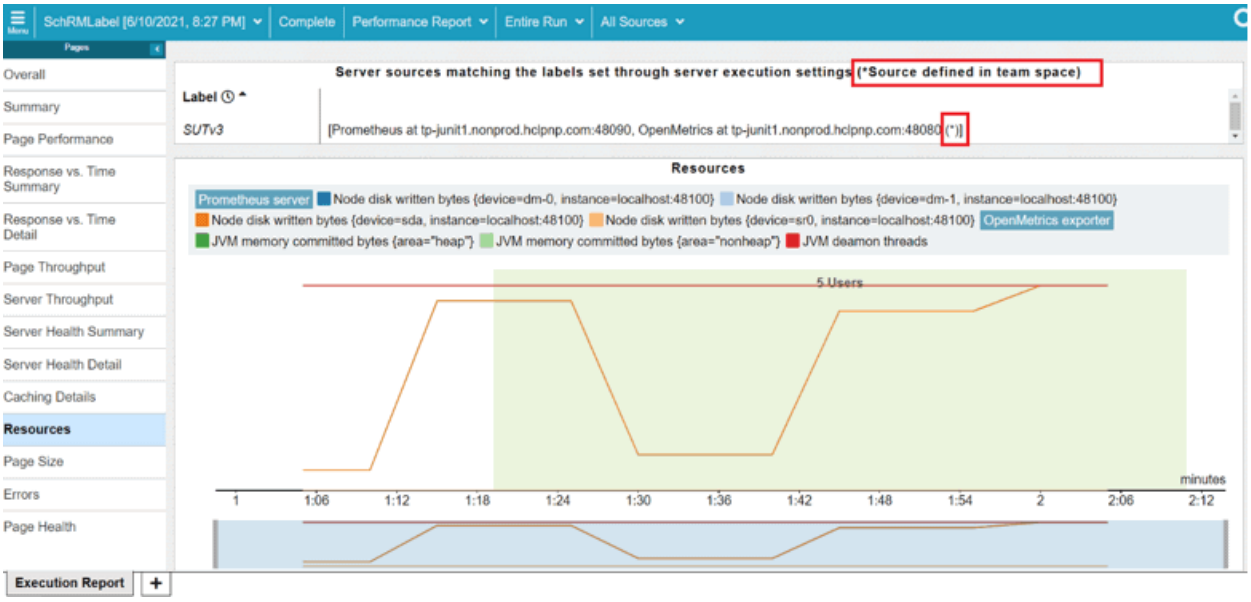
7. Select the labels that match the sources.
8. Click **Execute** to execute the performance schedule.
9. Click the **Result** view in the navigation bar of Test when the schedule is complete.
10. Expand the performance schedule result to see the resource monitoring label name in the **Details** card.

The **Details** card displays the resource monitoring labels that are used in the schedule to override the initial labels and collect resource monitoring sources (in the example, the 'SUTv3' label). For other details about the **Results** page, see the link at the end of this page.



In the **Report** card, you find the link to the Statistics Report that is generated. In the **Resource** tab of the Statistics Report, you can view the information related to the Server sources matching the labels set in the schedule.

11. Click the **Statistics Report** link in the **Reports** card to see the performance metrics of the monitoring sources matching the selected labels.



Example: You can see the statistics for the sources tagged with the 'SUTv3' label.

Note: In the Resources report, each resource name followed by an asterisk identifies a resource monitoring source that is defined in a team space.

Related information

Test results and reports on page 1501

Collecting and viewing resource monitoring usage data

With Test , you can collect resource monitoring data in a timeframe and view the resource monitoring usage data in a graph while the test is running. You can also create queries to retrieve the resource monitoring usage data.

About this task

While monitoring a system, you can view the current data in a graph but you might also want to view persistent data that is collected during a test run in a time frame. To view the persistent data, you can configure the test with additional parameters before the test execution. While the test is running, you can view the resource monitoring usage data in a graph. When the test is running or complete, you can create queries to retrieve the collected data results.

To collect and view resource monitoring data, perform these tasks:

1. Configure a test with two additional parameters in the advanced settings.
2. View the resource monitoring usage data in a graph.
3. Create queries to retrieve the resource monitoring usage data.

Related information

Configuring a test on page 1601

Viewing resource monitoring usage data on page 1603

Viewing resource monitoring usage data on Grafana on page 1605

Creating queries for resource monitoring usage data on page 1607

Configuring a test

You must configure a test with two parameters to collect resource monitoring usage data in a time frame when the test is running.

Before you begin

You must have completed the following tasks:

- Added and configured a source.
- Created resource monitoring labels for your source.

About this task

You can collect resource monitoring usage data during a test run for all types of monitoring sources, and all test types, except API stubs.

1. Click **Execution**, in the navigation pane.
2. Select the test that you want to run and click in the **Actions** column to open the **Execute test asset** window.
3. Click **Advanced settings**.
4. Enter the following parameters and values in **Additional Configuration Parameters**:

Parameter name	Value
rm.log.labels	Enter a label name in the second edit field as the parameter value. If you enter multiple labels, enter a comma to separate them in the list. **Example:** For two labels with the following names: 'LabelA' and 'LabelB': rm.log.labels LabelA,LabelB If a label contains a comma in its name, you must add two commas in order to collect the resource monitoring usage data. **Example:** For a label named 'Label1A,Label1B', enter the following parameter and value: rm.log.labels Label1A**,,**Label1B
rm.log.period.seconds	Enter a sampling period in the second edit field as the parameter value. The minimum period is one second.

5. Click **Add** for each parameter.
6. Click **Execute**.

Results

The monitoring data is collected while your test is running. You can see the collected data in a graph of the resource monitoring page, even 3 minutes after the test is complete.

What to do next

You can view the resource usage data on the **Resource Monitoring** page in Test and Grafana.

- Viewing resource monitoring usage data on page 1603
- Viewing resource monitoring usage data on Grafana on page 1605

Viewing resource monitoring usage data

After you add and configure monitoring sources in Test , you can view the resource monitoring usage data in a graph for each monitored source and for each selected counter.

Before you begin

You must have completed the following tasks:

- Added and connected a monitoring source, and an agent when required, to your project to be able to view the list of available sources and agents, and the resource monitoring usage data in your project.
- Configured a test with advanced parameters to view persistent usage data in a graph.

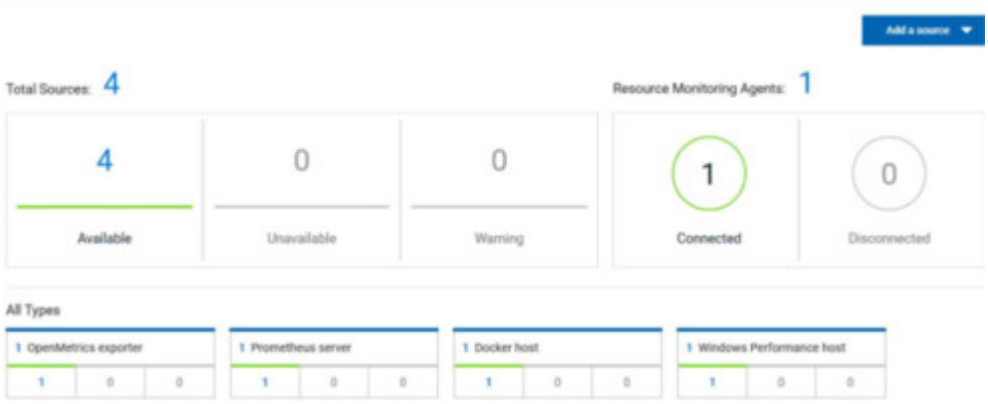
About this task

After you add a monitoring source to Test, configure the source and select monitoring counters, you can view the current resource monitoring usage data for each selected counter in a graph. If you configure the test to collect data, you can also view the resource monitoring usage data that is collected for a defined timeframe. In this case, the data is still visible in the graph three minutes after the test run is complete. After this time, you can create queries to retrieve the data.

Note: For a Prometheus source, as data comes from the Prometheus database, it no longer shows up in the graph when the test run is complete. However, the data is collected and shows up in the queries if it matches a label.

1. Click a link either in the **Total Sources** cards or in the **All Types** cards from the resource monitoring page to access the resource monitoring usage data.
   The tables display the total number of sources in your project and the number of sources by type. You can also view the number of available and unavailable sources, and the number of sources that contain warnings. If you have added '4' sources to your project, for example, click '4' in the table to access the page where you can see the resource usage data for each source.

   If you have added agents to your project, you can also view the number of agents that are connected and disconnected to your project.

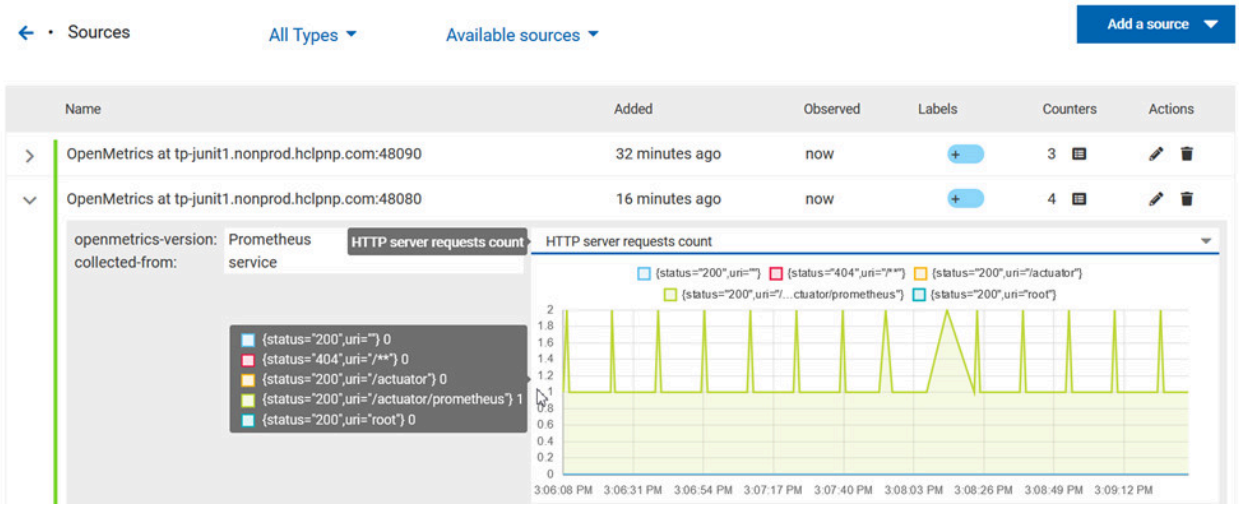


Result

The list of sources is displayed in a new page.

You can filter the list of sources by collector type or by status.

2. Expand a monitoring source under the **Name** column to see the current monitoring data usage in a graph.
3. Click the down arrow to select a counter in the list.
   A graph is displayed for each counter.



What to do next

From the page where you can view the resource monitoring usage data, you can also manage your sources. See Managing sources on page 1593.

Related information

Collecting and viewing resource monitoring usage data on page 1601

Creating queries for resource monitoring usage data on page 1607

Viewing resource monitoring usage data on Grafana

After you add and configure monitoring sources in Test , you can view the resource monitoring usage data in a graph for each monitored source and for each of the selected counters on Grafana.

Before you begin

You must have completed the following tasks:

- Added and connected a monitoring source, and an agent when required, to your project to view the list of available sources and agents, and the resource monitoring usage data in your project.
- Configured a test with advanced parameters to view persistent usage data in a graph. See Configuring a test on page 1601.
- Installed the DevOps Test Hub Results Data plug-in and created an instance of data source for DevOps Test Hub Results Data. See Installing and configuring the DevOps Test Hub Results Data plugin on page 1516.
- Opened Grafana with one of these ports: `http://localhost:3000` or `http://localhost:8080`.
- Started the test execution on Test.

About this task

After you add a monitoring source to Test, you must configure the source, and select monitoring counters. You can view the resource monitoring usage data for each of the selected counters in a graph on Grafana. You must configure the test to collect and view the resource monitoring usage data that is collected for a defined timeframe. During the test execution, the data can be seen with the default time range on Grafana. After the test execution is complete, you must carefully select the time range so that it corresponds to the test execution start and end time.

1. Click **Explore** in the **Toggle** menu in Grafana.

   The **Data query** page with the default data source is displayed.
2. Select **DevOps Test Hub Results Data** from the list of data sources.
3. Select the type of query by performing any of the following actions when you want to view statistics from the test data:

To view	Actions
The variation of a metric over time.	Perform the following steps: a. Click the time-series statistics tab icon . b. Go to Step 4 on page 1605.
The values of the metric that are computed over the entire time range.	Perform the following steps: a. Click the overall statistics tab icon . b. Go to Step 4 on page 1605.

4. Define the scope of the query by performing the actions indicated in the following table:

Options	Actions
If you want to use the data from the team space	Perform the following steps: a. Select the **Team Space** option. b. Select the team space from the list of team spaces that are listed. c. Select **Resource Monitoring** from the list in the **Service** option. d. Go to Step 5 on page 1606.
If you want to use the data from your project	**Note:** You can select the project as the scope for your query if you want to view any of the supported query types. Perform the following steps: a. Select the **Project** option. b. Select the project from the list of projects that are listed. c. Select the **Service** option. d. Select **Resource Monitoring** from the list e. Go to Step 5 on page 1606.

5. Optionally, enter a name for the metric that helps you identify it in the **Metric** field.
6. Select a metric from the list of metrics in the row of the **Metrics** option.
7. Select the aggregation factor for the metric. For example, select **Sum** to add all values of the metric.
8. Optionally, click the **Cumulated** toggle button to add the values for the metric cumulatively over time.

   Note: The **Cumulated** is set to off. This is the default selection.

9. Add another metric by clicking the **Add metric** icon .

   Note: If you added multiple metrics and you want to remove a metric, you can click the **Remove metric** icon of the metric.

10. Click **Add query** if you want to combine queries from different projects, results ids, or with different properties in the query filters.
11. Click **Run query**.

    Result

    A graph is displayed for the selected metrics.

Results

You viewed the resource monitoring usage data in a graph for each monitored source and for each of the selected counters on Grafana.

What to do next

You can build a dashboard including graphs, charts, gauges, and tables to get customized visual representations of your resource monitoring usage data. See Dashboard overview.

Creating queries for resource monitoring usage data

After you execute a test to collect resource monitoring usage data, you can create queries in Test to retrieve the data.

Before you begin

You must have added resource monitoring sources to a project.

About this task

You can no longer view the monitoring usage data in graphs 3 minutes after completion of the test. However, you can create queries to retrieve the list of activities and events in your tests. You can use the following procedure as an example to create your own queries.

1. Follow these steps to filter the list of activities that are collected for a duration of less than 300 seconds:
    a. Click **Activities**.
    b. Enter `'Type is RMsource and duration < 300'` in the **Query filters** field.
    c. Enter `'time ASC'` in the **Sort by** field to sort data by ascending order.
    d. Click **Run Query**.
    e. Click **Save as**.
    f. Enter a name to identify the new query and click **Save**.

    The list of activities that are collected for a duration of less than 300 seconds is displayed in the query result table, and this list is sorted in an ascending order.
2. Follow these steps to filter the list of observations that are performed during a test execution and defined by the test result identifier:
    a. Click **Events**.
    b. Enter the following filters in the **Query filters** field:

    ```
    type is observation and anyParentActivity matches ( anyChildActivity matches (requesterNature = 'test-result' and requesterId=1202 ))
    ```

    c. Enter `'time ASC, parentActivity.description ASC, specifier ASC'` in the **Sort by** field to sort data by ascending order.

    Note: You can use the **Ctrl-Space** keyboard shortcut to enter the properties and operators.

    d. Click **Run Query**.
    e. Click **Save as**.
    f. Enter a name to identify the new query, and then click **Save**.

    The new query results are displayed in the table of activities and events.
3. Follow these steps to manage your query:

- Click the action menu that is next to the **Save** button and select **Reset** if you have just modified a query and you want to undo your modification.
- Click the action menu and select **Delete** to delete the new query from the **Events and activities** queries list.

4. Follow any of these steps to configure the query result table:
   - Click the action menu in a column of the table to change the column display, and then perform any of the following actions:
     - Click **Insert new column**, enter a field name in **Search new column** and click **Add**.
     - Click **Move** to move a column in the table.
     - Click **Remove** to remove a column from the table.
   - Follow these steps to add a column at the end of the table:
     a. Click **Add column**.

     Result: A window opens.

     b. Enter a field name in **Search new column**.
     c. Click **Add**.
5. Click **Save** to save the configuration of the results table.

Results

You have created two queries in your project to display:

- The list of activities that are collected for a duration of less than 300 seconds.
- The list of observations that are performed during a test execution and defined by the test result identifier.

Related information

Configuring and running log queries on page 1539

Configuration of a change management

When you use any application to create and track defects (bugs), issues, or other work items, you can configure the application as a change management on Test . You can then create defects (bugs), issues, or other work items without the need to open your application.

You use Test to run tests for the software that you develop and view their results. When you discover that you might want to raise defects (bugs), issues, or other types of work items for these tests, you are required to open the application that you use. You can only then, create defects (bugs), issues, or other work items in your application.

When you configure your application as a change management on Test, you can create defects (bugs), issues, or other work items without the need to open your application.

You can find details in the following table about the usage of the applications that you can configure as a change management on Test.

Note: You can configure only one application as the change management on Test.

Issue tracking application	Usage as a change management	Go to...
Atlassian Jira Software	You can create stories, defects, or tasks as issues in your project in Jira from Test and also launch your project in Jira from Test to track and manage these issues.	Configuration of Atlassian Jira as a change management on page 1609
Azure DevOps	You can create work items in your project in Azure DevOps from Test and also launch your project in Azure DevOps from Test to track and manage these work items.	Configuration of Azure DevOps as a change management on page 1621
DevOps Plan	You can use Plan as a change management in Test. You can also launch Plan from Test to track and manage change requests and other records.	Configuration of DevOps Plan as a change management on page 1630

Configuration of Atlassian Jira as a change management

You can use Atlassian Jira as a change management in Test to create issues in the Jira project for the tests that ran in Test. The links to the Jira issues are displayed on the **Results** page in Test, and you can then view and track the Jira issues in Jira.

Prerequisites

Before you configure Jira as a change management on Test, you must have completed the following tasks:

- Installed Jira Software. See Supported installations on page 1609.
- Ensured that you obtained the access and authentication details to Jira from the Jira administrator.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

Supported installations

The following types of Jira Software installations are supported that can be configured as a change management on Test:

- On-premise Jira Software installations.
- On-cloud Jira Software installations.

Task flow

You can follow the tasks that are listed in the following table when you want to use Jira as a change management on Test. The table also provides you the links to the information about the tasks.

Task	Go to...
Configure Jira as a change management on Test.	Configuring Jira as a change management on page 1610
Create defects or issues for tests that ran on Test.	Creating Jira issues for the test results on page 1612
View details of the Jira issues that you created from Test.	Viewing details of the Jira issues on page 1618

You can manage the Jira settings that you configured in Test. You can edit the saved configuration or you can delete the configuration. For more information, see Managing the Jira configuration on page 1620.

Configuring Jira as a change management

You must configure Jira as a change management in your project in Test so that can create issues in the Jira project for the tests that ran.

Before you begin

You must have completed the following tasks:

- Verified that you can access Jira.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

About this task

With the introduction of OAuth 2.0 authentication for the connection between Test and Jira, you must be familiar with the creation of Apps that are based on the OAuth 2.0 integration. There can be more than one method in the Atlassian Developer console that you can configure the created App. One such method is adopted in the configuration of an App in Jira.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Click **Add Change Management Integration** option within the **Change Management** section.
5. Perform the following steps in the **Add Jira configuration** dialog:

a. Enter the URL of the Jira server or Jira cloud in the **Jira URL** field, and then click **Next**.

   Test provides the *Callback URL* that you must use to generate the Client ID and Client secret for the App in Jira.

b. Click the **Atlassian Developer console** link.

   The **Atlassian Developer console** page is displayed.

   Perform the following steps on the **Atlassian Developer console** page:

   i. Click **Create > OAuth 2.0 integration**.
   ii. Enter a name for the App in the **Name** field.
   iii. Click the **I agree to be bound** checkbox, and then click **Create.**

      The App is created in the Atlassian Developer Console and the details are displayed for the App.

   iv. Click **Authorization** in the navigation, and then click **Add** in the row of OAuth 2.0 (3LO).

      The OAuth 2.0 configuration page is displayed. You must paste the *Callback URL* provided by Test.

   v. Copy the *Callback URL* from the **Add Jira configuration** dialog in Test and then paste it into the **Callback URL** field.
   vi. Click **Save changes**.

      The App in Jira is configured to authenticate the requests from Test by using the OAuth 2.0 protocols.

   vii. Click **Permissions** and then, click the **Configure** option in the row of **Jira API**.
   viii. Click **Edit Scopes** on the **Jira API** page, and then select the following scopes for your App that are required for a successful connection with Test:
      - View Jira issue data
      - Manage Jira global settings
      - View user profiles
      - Create and manage issues

   ix. Click **Save** to save the App settings.
   x. Click **Settings** and scroll down the page to view the **Authentication details** panel.

      The **Client ID** and **Secret** must be copied from the **Settings** page to the corresponding fields on the **Add Jira configuration** dialog.

c. Click the **Copy** icon in the **Client ID** field to copy the key, and then paste the copied key in the **Client ID** field on the **Add Jira configuration** dialog.

d. Click the **Copy** icon in the **Secret** field to copy the key, and then paste the copied key in the **Secret** field on the **Add Jira configuration** dialog.

6. Click **Save** on the **Add Jira configuration** dialog.

You added the URL of the Jira cloud and configured an App in Jira to authenticate the connection with Test by using OAuth 2.0. You can view that the URL of the Jira server or cloud instance is displayed within the **Change Management** pane.

7. Click the **Configure** icon .

   A page in Atlassian is displayed in a browser that states that the App you created previously is requesting access to your Atlassian account.

8. Read the message that is displayed, and then click **Accept**.

   When the connection between Test and the App in Jira is established successfully, the **Configure Jira project** dialog is displayed with the Jira URL that is configured as a change management system in Test.

9. Select the project in Jira from the list under which you want to create defects.
10. Click **Save**.

Results

You configured Jira as a change management system in Test.

What to do next

You can perform the following tasks:

- View the following configuration details of Jira on the **Change Management** about the Jira configuration:
  - The URL of the Jira server or cloud instance.
  - The project that you selected.
- Create issues for the test results that are available in your project in Test. See Creating Jira issues for the test results on page 1612.

Creating Jira issues for the test results

You can create Jira issues for the tests that ran in Test . The issues are created in the project in Jira that you configured in Test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Configured Jira as a change management for your project in Test. See Configuring Jira as a change management on page 1610.
- Configured a test run. See Test run configurations on page 1134.
- Viewed the results of the completed test runs on the **Results** page. See Test results on page 1499.

About this task

When you want to create a Jira issue in Test, the issue types that display in the **Create an issue in Jira** dialog depend on the configuration done by the administrator of Jira.

The default fields that display on the **Create an issue in Jira** dialog are as follows:

- **Issue Type**
- **Summary**
- **Description**
- **Attach Reports**

If you want to view any additional fields in the **Create an issue in Jira** dialog from Test, then you must inform your Jira administrator to configure those additional fields as the required fields in the project in Jira. You can then use those fields in the **Create an issue in Jira** dialog when you create a defect from Test.

You can refer to the **Related information** section to know more about Jira field configuration.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

   Note: You can click the relevant tab to view the tests that were run earlier.

4. Perform any of the following actions to identify the test asset:

◦ Scroll through the list.

Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	VU Schedule			

- Search for the test by entering any text that is contained in the test asset name in the **Search** field.
- Select the time interval within which the test was run.

  For example, you can select the *Last 24 hours* option to view the tests that were run in the previous 24 hours.
- Select the range of dates within which the test was run.

  You can click the calendar icon to select the date in the first image as the start date, and then select the date in the second image as the end date.

  Note: You can select the range of dates only when you select the time interval as *Custom*.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab, and then select the test type.

  All the tests based on the selected test type are displayed.

  For example, you can select **API Suite** in the **SUITES** tab to display all the API suites or you can select **Functional Test** in the **TESTS** tab to display all the functional tests.
- Click the **Filter** icon in the **ADVANCED** tab, and then perform the following actions to create a filter query:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then select **Add rule** to add a rule or **Add group** to add a group of rules.
  d. Select or enter the relevant parameters, and then either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - ID
     - Type
     - Test
     - Version
     - Result
     - Started on
     - Finish on
     - Duration

- Status
- Verdict
- Labels
- Locked
- Repository
- Branch
- Text

e. Click **Save**.

The filter query is saved and all the tests that match the filter criteria are displayed.

- Perform the following actions in the **ADVANCED** tab to retrieve and apply a previously saved filter query:

a. Click the **Filter** icon .

All the filter queries that you created and saved earlier are displayed.

b. Click the filter query that you want to apply.

All the tests that match the filter criteria are displayed.

You have identified the test that you want to use.

5. Select the step that depends on the role assigned to you in the project on Test:

- If you are a member with a role assigned as *Owner*, go to Step 6 on page 1616.
- If you are a member with a role assigned as *Tester*, and if you did not complete the one-time authorization process, go to Step 7 on page 1616.
- If you are a member with a role assigned as *Tester* and completed the one-time authorization process, go to Step 8 on page 1617.

6. Perform the following steps if you are assigned the role of a project *Owner*:

a. Click the **Jira: Create a Jira issue** icon that is under the **Actions** column in the row of the test.

Alternatively, you can click the test under the **Test** column to view the results panel, and then click the **Jira: Create a Jira issue** icon that is displayed in the panel.

The **Create an issue in Jira** dialog is displayed.

b. Go to Step 9 on page 1617.

7. Perform the following steps if you are assigned the role of a *Tester* and if you have not yet completed the one-time authorization process:

Note: You must complete the one-time authorization process if you are not a project owner.

a. Click the **Jira: Create a Jira issue** icon that is under the **Actions** column in the row of the test.

   Alternatively, you can click the test under the **Test** column to view the results panel, and then click the **Jira: Create a Jira issue** icon that is displayed in the panel.

   The **Configure Jira server to authorize the application** dialog is displayed.

b. Click the URL of the Jira server.

c. Log in to the Jira server account with your valid credentials.

   Note: The **Login** page is displayed only if you are not already logged in to Jira.

d. Read the message that is displayed, and then click **Allow** to authorize the Jira account.

   After the authorization is successful, the **Access Approved** page is displayed with a verification code.

e. Copy the verification code, and then enter the verification code in the field on the **Configure Jira server to authorize the application** dialog of Test.

f. Click **Save**.

g. Go to Step 8 on page 1617.

8. Perform the following steps if you are assigned the role of a *Tester* and you have completed the one-time authorization process:

   a. Click the **Jira: Create a Jira issue** icon that is under the **Actions** column in the row of the test.

      Alternatively, you can click the test under the **Test** column to view the results panel, and then click the **Jira: Create a Jira issue** icon that is displayed in the panel.

      The **Create an issue in Jira** dialog is displayed.

   b. Go to Step 9 on page 1617.

9. Perform the following steps in the **Create an issue in Jira** dialog:

   Note: The fields that are displayed might depend on the fields configured in the Jira project.

   a. Select the issue type that you want to create.

   b. Enter text for the title of the Jira issue. The name of the test is displayed as the default text in the **Summary** field.

You can view the details of the test run that is displayed in the **Description** field. These details are appended to the Jira issue under the **Description** field, after you create the issue.

c. Select the existing labels in your Jira project or enter a new name for the label in the **Label** field, if required.

Note:

If the **Label** field is set to "Required" in your Jira project, you must select an existing label or create a new label by providing a name in the **Label** field. If you don't select an existing label or create a new label, an error message is displayed that indicates the **Label** field is a required field.

d. Select the test reports or test logs of the test run that you want to attach to the Jira issue.

Note: You can select any report, test log, or all the reports and the test log as attachments.

e. Click **Create**.

The issue is created in the project on the Jira server and the link to the issue is displayed in the **Results** panel of the test on Test.

Note: You can create multiple issues for the same test result, if you want.

Results

You created issues in the Jira project for the test results from Test.

The Jira issues are created in the Jira project and the links to the issues are displayed in the **Details** panel of the test result on the **Results** page.

What to do next

You can perform any of the following tasks:

- View details of the Jira issues that you created in Jira for the test. See Viewing details of the Jira issues on page 1618.
- Modify the configuration that you set for Jira on Test. See Managing the Jira configuration on page 1620.

Related information

Specifying Jira fields

Viewing details of the Jira issues

After you create Jira issues in Test , you can view and track the details of the Jira issues in Jira.

Before you begin

You must have completed the following tasks:

- Completed a test run and the test results are available on the **Results** page.
- Created Jira issues against the test results. See Creating Jira issues for the test results on page 1612.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

   Note: You can click the relevant tab to view the tests that were run earlier.

4. Click the test that is listed under the **Test** column.
5. Click the **Report** tab in the **Results** panel, if it is not already displayed.
6. Click the Jira issue link under **Defect tracking**.

   The Jira log-in page is displayed, if you are not already logged into Jira.
7. Enter your username and password.
8. Click **Login**.

   The issue details are displayed in Jira.

   Note: The **Description** field of the Jira issue in Azure DevOps provides additional details of the test that ran in Test such as the name of the test, the person who ran the test, the verdict of the test result, the date and time

of the test run, the links to the test report that is generated in Test, and the link to the query that is created. You can also view the reports that you attached as a compressed file under the **Attachments** field in the Jira issue.

Results

You have viewed the details in Jira of the Jira issue that you created in Test.

Managing the Jira configuration

You can manage the Jira configuration that you set in Test when you want to modify the project configured in Jira or when you want to delete the existing Jira configuration.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you obtained the access and authentication details to Jira from the Jira administrator.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Go to the **Configuration details of Jira** within the **Change Management** section, the configured details of the Jira integration are displayed.
5. Perform any of the following actions when you want to update or delete the Jira server configuration:
   - To update the project in Jira that was set previously in the Jira configuration. Go to Step 6 on page 1620.
   - To delete the Jira configuration. Go to Step 7 on page 1620.
6. Perform the following steps to modify the project in Jira:

   a. Click the **Configure** icon of the Jira integration that you want to manage.
      The **Select project in Jira** dialog is displayed.

   b. Select the project that you want from the list displayed in the **Project** field.

   c. Click **Update** to save the updated project in the Jira server configuration.

7. Perform the following steps to delete the existing Jira server configuration:

a. Click the **Delete** icon .

The **Delete Jira configuration** dialog is displayed.

b. Click **Confirm**.

The Jira configuration is deleted and a confirmation message is displayed.

Note: The links to the issues that you created for the tests that ran are not deleted from the results panel in Test even after you delete the Jira configuration.

Results

You achieved the following results:

- Updated the project in Jira and saved the Jira configuration.
- Deleted the Jira configuration.

Configuration of Azure DevOps as a change management

You can use Azure DevOps as a change management in Test to create work items in the Azure DevOps project for the tests that ran in Test. Links to the Azure DevOps work items are displayed on the **Results** page in Test, and you can then view and track the work items in the Azure DevOps server.

Prerequisites

Before you configure Azure DevOps as a change management in Test, you must have completed the following tasks:

- Ensured that you obtained the access and authentication details to Azure DevOps from the Azure DevOps administrator.
- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

Supported installations

The following types of Azure DevOps installations are supported that can be configured as a change management system on Test:

- On-premise Azure DevOps installations.
- On-cloud Azure DevOps installations.

Task flow

You can follow the tasks that are listed in the following table when you want to use Azure DevOps as a change management in Test. The table also provides you the links to the information about the tasks.

Task	Go to...
Configure Azure DevOps as a change management.	Configuring Azure DevOps as a change management on page 1622
Create Azure DevOps work items for the test asset.	Creating Azure DevOps work items on page 1623
Track and view the details of the created Azure DevOps work items.	Viewing details of the Azure DevOps work items on page 1628

You can edit the saved configuration, or you can delete the configuration and create a new configuration. For more information, see Managing the Azure DevOps configuration on page 1629.

Configuring Azure DevOps as a change management

You must configure Azure DevOps as a change management in your project in Test before you can create work items in the Azure DevOps project for the tests that ran.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that you obtained the access and authentication details to Azure DevOps from the Azure DevOps administrator.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Click the **Add Change Management Integration** option, and then select **Azure DevOps** option from the list.

   The **Configure Azure DevOps** dialog is displayed.
5. You can perform the following actions:

   a. Enter the URL of **Azure DevOps**.

   Note: You must enter the URL of the Azure DevOps in one of the following formats:
   - https://dev.azure.com/{Organization}
   - https://{Organization}.visualstudio.com

- https://{Hostname}:{Port}/{Collection}

  The URL contains information such as the server name (hostname, port, or domain name) and organization name.

b. Click **Next**, and then paste the copied personal access token.

Note: If you do not have your personal access token, you can click **Generate** to generate your personal access token to authorize full access or access to the work items.

c. Click **Next**, and then select an **Azure DevOps** project from the drop-down list.

d. Click **Add**.
The configuration is added and saved within the **Change Management** section under **Configure details of Azure DevOps.**

Results

You configured Azure DevOps as a change management in Test.

What to do next

You can create work items for the test results that are available in your project in Test. See Creating Azure DevOps work items on page 1623.

Creating Azure DevOps work items

You can create Azure DevOps work items for the tests that ran in Test . The issues are created in the project in Azure DevOps that you configured in Test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Configured Azure DevOps as a change management system in Test for your project. See Configuring Azure DevOps as a change management on page 1622.
- Extended the CA list. See Extending the trusted certificate authority list on Linux on page 735 or Extending the trusted certificate authority list on Windows on page 736.
- Configured a test run. See Test run configurations on page 1134.
- Viewed the results of the completed test runs on the **Results** page. See Test results on page 1499.

About this task

When you want to create a Azure DevOps work item on Test, the fields that display in the **Create a work item** dialog depends on the configuration done by the Azure DevOps administrator.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

   Note: You can click the relevant tab to view the tests that were run earlier.

4. Perform any of the following actions to identify the test asset:
   - Scroll through the list.

     Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

     Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test by entering any text that is contained in the test asset name in the **Search** field.
- Select the time interval within which the test was run.

  For example, you can select the *Last 24 hours* option to view the tests that were run in the previous 24 hours.
- Select the range of dates within which the test was run.

  You can click the calendar icon to select the date in the first image as the start date, and then select the date in the second image as the end date.

Note: You can select the range of dates only when you select the time interval as *Custom*.

- Click the **Filter** icon in the **SUITES** or **TESTS** tab, and then select the test type.

  All the tests based on the selected test type are displayed.

  For example, you can select **API Suite** in the **SUITES** tab to display all the API suites or you can select **Functional Test** in the **TESTS** tab to display all the functional tests.

- Click the **Filter** icon in the **ADVANCED** tab, and then perform the following actions to create a filter query:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then select **Add rule** to add a rule or **Add group** to add a group of rules.
  d. Select or enter the relevant parameters, and then either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - ID
     - Type
     - Test
     - Version
     - Result
     - Started on
     - Finish on
     - Duration
     - Status
     - Verdict
     - Labels
     - Locked
     - Repository
     - Branch
     - Text

  e. Click **Save**.

     The filter query is saved and all the tests that match the filter criteria are displayed.

- Perform the following actions in the **ADVANCED** tab to retrieve and apply a previously saved filter query:
  a. Click the **Filter** icon .

     All the filter queries that you created and saved earlier are displayed.

  b. Click the filter query that you want to apply.

     All the tests that match the filter criteria are displayed.

You have identified the test that you want to use.

5. Perform the following actions on the test that was run to create a work item:

   a. Click the **Azure: Create work item** icon that is under the **Actions** column in the row of the test.

      Alternatively, you can click the test under the **Test** column to view the results panel, and then click the **Azure: Create work item** icon that is displayed in the panel.

      The **Create a work item** dialog is displayed.

   b. Perform the actions indicated in the following table for each of the fields:

Field	Description	Action
Project	The name of the Azure DevOps project.	Select the Azure DevOps project from the drop-down list.
Work item type	The type of work item.	Select the work item you want to assign for the test asset.
Title	A title that identifies the work item.	Specify a title for the work item.
Description	A detailed description of the work item.	Enter a description of the work item.

      Note: The other fields selected as **Required** fields on the **Azure DevOps** project are displayed in the **Create a work item** dialog.

   c. Enter the information for all the required fields, and then click **Create**.

      Note: You can create multiple work items for the same test result, if required.

Results

You created work items in the Azure DevOps project for the test results from Test.

The work items are created in the Azure DevOps project and the links to the work items are displayed in the **Details** panel of the test result on the **Results** page.

What to do next

You can perform any of the following tasks:

- View details of the Azure DevOps work items that you created in Azure DevOps for the test. See Viewing details of the Azure DevOps work items on page 1628.
- Modify the configuration that you set for Azure DevOps on Test. See Managing the Azure DevOps configuration on page 1629.

Viewing details of the Azure DevOps work items

After you create a work item in Test , you can view and track the details of the Azure DevOps work item in Azure DevOps.

Before you begin

- Created Azure DevOps work item against the test results. See Creating Azure DevOps work items on page 1623.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

   Note: You can click the relevant tab to view the tests that were run earlier.

4. Click the test that is listed under the **Test** column.
5. Click the **Report** tab in the **Results** panel, if it is not already displayed.
6. Click the Azure DevOps work item link under **Defect tracking**.

   The Azure DevOps log-in page is displayed.
7. Enter your username and password.
8. Click **Login**.

The work item details are displayed in Azure DevOps.

Note: The **Description** field of the work item in Azure DevOps provides additional details of the test that ran in Test such as the name of the test, the person who ran the test, the verdict of the test result, the date and time of the test run, the links to the test report that is generated in Test, and the link to the query that is created.

Results

You have viewed the details in Azure DevOps of the Azure DevOps work item that you created in Test.

Managing the Azure DevOps configuration

You can edit or modify the Azure DevOps configuration that you added and saved to the Change management section on Test . If you do not want to use the existing configuration and want to set up a new Change management configuration, then you can delete the configuration.

Before you begin

You must have completed the following task:

- Been assigned a project *Owner* role.

1. Open the project that you created on Test.
2. Go to **Manage**, and then click **Integration** in the navigation pane.
   The **Integration** page is displayed.
3. Go to the **Configuration details of Azure Devops** within the **Change Management** section, the configured details of the Azure DevOps integration are displayed.
4. Perform any of the following actions:
   - To edit the Azure DevOps configuration. Go to Step 5 on page 1629.
   - To delete the Azure DevOps configuration. Go to Step 6 on page 1630.
5. Follow the step to edit the Azure DevOps configuration:

   a. Click **Edit** icon of the Azure DevOps integration that you want to manage.
      The **Configure Azure DevOps** dialog is displayed.

   b. Edit the URL by performing the following steps:
      i. Enter the URL of **Azure DevOps**.
      ii. Click **Next**.
      iii. Paste the copied personal access token or click **Generate** to generate your personal access token.
      iv. Click **Add**.

      The URL is updated and saved.

   c. Edit the project by performing the following steps:

i. Select an **Azure DevOps** project from the drop-down list.
ii. Click **Add**.

The project is updated and saved.

6. Follow these steps to delete the Azure DevOps configuration:

a. Click the **Delete** icon .
The **Delete Azure DevOps configuration** dialog is displayed.

b. Click **Delete**.
The Azure DevOps configuration is deleted.

Results

You achieved the following results:

- Edited the Azure DevOps configuration and saved the configuration.
- Deleted the Azure DevOps configuration.

Configuration of DevOps Plan as a change management

You can use HCL DevOps Plan (Plan) as a change management to create work items after you run tests on Test . You can then view and track the work items in Plan.

Prerequisites

Before you configure Plan on Test, you must have completed the following tasks:

- Ensured that you obtained the access details and authentication details of Plan from the Plan administrator.
- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

Task flow

You can follow the tasks that are listed in the following table when you want to use Plan as a change management on Test. The table also provides links to the information about the tasks.

	Task	Go to...
1	Configure Plan as a change management.	Configuring DevOps Plan as a change management on page 1631
2	Create work items in Plan for the tests that ran on Test.	Creating work items on page 1632

	Task	Go to...
3	View and track the details of the created work items.	Viewing details of the work items on page 1637

You can manage the Plan settings that you configured in Test. You can edit the saved configuration or you can delete the configuration. For more information, see Managing the DevOps Plan configuration on page 1638.

Configuring DevOps Plan as a change management

You can configure HCL DevOps Plan (Plan) as a change management system in your project to create work items from Test in Plan.

Before you begin

You must have completed the following tasks:

- Read about the prerequisite tasks in Prerequisites on page 1630.
- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you obtained the access details and authentication details of Plan administrator.
- Created a project in your team space in Test. See Creating a project on page 1728.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Click **Add Change Management Integration** option within the **Change Management** section.
5. Select the **DevOps Plan** option from the list.

   The **Configure DevOps Plan application** dialog is displayed.
6. Configure the Plan application by performing the following steps:

   a. Enter the URL of the Plan server.

   Note: You can enter the URL of Plan in any of the following formats:
   - https://*IP_address*:*port*/
   - https://*hostname*/

Where	Is
IP_address	The IP address of the server on which Plan is installed.

Where	Is
port	The port that is used to communicate with Plan .
hostname	The hostname of the computer on which Plan is installed.

b. Enter the ID for the Plan Teamspace.

c. Enter the name of the Plan application.

d. Click **Add**.

The configuration is added and saved within the **Change Management** section under the **Configuration details of DevOps Plan**.

Results

You have configured Plan as a change management in your project on Test.

What to do next

You can perform any of the following tasks:

- Create work items in Plan for the tests that you ran on Test. See Creating work items on page 1632.
- Modify the configuration that you set for Plan on Test. See Managing the DevOps Plan configuration on page 1638.

Creating work items

You can create work items in HCL DevOps Plan (Plan) for the tests that ran on Test . For example, you might want to create a work item in Plan when the test results differ from the expected results.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Configured Plan as a change management in Test. See Configuring DevOps Plan as a change management on page 1631.
- Configured applications in Plan.
- Configured a test run. See Test run configurations on page 1134.
- Viewed the results of the completed test runs on the **Results** page. See Test results on page 1499.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

The **Overview** page of the project is displayed.

3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

   Note: You can click the relevant tab to view the tests that were run earlier.

4. Perform any of the following actions to identify the test asset:

   - Scroll through the list.

     Tip: You can hover over the icon in the **Type** column to know the type of the test asset.

     Note: You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	API test		✓	✓
	Functional test		✓	✓
	Performance test		✓	✓
	AFT Suite	✓		✓
	API Suite	✓		✓

Icon	Represents	Listed in the SUITES tab	Listed in the TESTS tab	Listed in the ADVANCED tab
	Compound Test	✓		✓
	HCL® AppScan CodeSweep	✓		✓
	JMeter Test	✓		✓
	JUnit Test	✓		✓
	Postman test	✓		✓
	Rate Schedule	✓		✓
	Selenium test	✓		✓
	Server Suite asset	✓		✓
	Server Test asset		✓	✓
	VU Schedule	✓		✓

- Search for the test by entering any text that is contained in the test asset name in the **Search** field.
- Select the time interval within which the test was run.

  For example, you can select the *Last 24 hours* option to view the tests that were run in the previous 24 hours.
- Select the range of dates within which the test was run.

  You can click the calendar icon to select the date in the first image as the start date, and then select the date in the second image as the end date.

  Note: You can select the range of dates only when you select the time interval as *Custom*.
- Click the **Filter** icon in the **SUITES** or **TESTS** tab, and then select the test type.

  All the tests based on the selected test type are displayed.

For example, you can select **API Suite** in the **SUITES** tab to display all the API suites or you can select **Functional Test** in the **TESTS** tab to display all the functional tests.

- Click the **Filter** icon in the **ADVANCED** tab, and then perform the following actions to create a filter query:
  a. Click **New filter**.
  b. Enter a name for the filter.
  c. Select an operator, and then select **Add rule** to add a rule or **Add group** to add a group of rules.
  d. Select or enter the relevant parameters, and then either select or enter the condition and the criteria for the condition.

     You can select a parameter from the following list:
     - ID
     - Type
     - Test
     - Version
     - Result
     - Started on
     - Finish on
     - Duration
     - Status
     - Verdict
     - Labels
     - Locked
     - Repository
     - Branch
     - Text
  e. Click **Save**.

     The filter query is saved and all the tests that match the filter criteria are displayed.
- Perform the following actions in the **ADVANCED** tab to retrieve and apply a previously saved filter query:
  a. Click the **Filter** icon .

     All the filter queries that you created and saved earlier are displayed.
  b. Click the filter query that you want to apply.

     All the tests that match the filter criteria are displayed.

You have identified the test that you want to use.

5. Perform the following actions on the identified test to create a work item in Plan:

a. Click the **DevOps Plan: Create a work item** icon that is under the **Actions** column in the row of the test.

   Alternatively, you can click the test under the **Test** column to view the results panel, and then click the **DevOps Plan: Create a work item** icon that is displayed in the panel.

   The **Create a work item in DevOps Plan** dialog is displayed.

b. Select the application from the list in the **Application** field.

   Remember: The application that you select must be the application in Plan for which you ran the test.

c. Enter the username in the **Username** field.

   Note: The username is authorized to access the Plan server.

d. Enter the password in the **Password** field to authenticate the username.

   Note: After the credentials that you provided in the **Username** and **Password** fields are authorized by the Plan server, you need not enter these values for subsequent connections to create a work item.

e. Click **Next**.

   The dialog displays the URL of the Plan server and the **Application** that you selected.

f. Select the work item type as *WorkItem*.

g. Click **Next**.

h. Select the project from the list in the **Project** field.

i. Select the type of the work item from the list in the **Type** field.

j. Enter text that is to be used as the title of the work item in the **Title** field. The text entered is displayed as the title in the work item in Plan.

   You can view the details of the test run in the **Description** field, which is appended to the work item that is created for the test run in Plan.

k. Click **Create**.

A work item is created in Plan and a link to the work item is created in the **Report** panel of the **Results** page.

Notes:

- You cannot delete the work item that is created in Plan from Test.
- You can create multiple work items for the same test result, if required.

Results

You have created a work item in Plan from the **Results** page for the test that ran on Test.

What to do next

You can perform any of the following tasks:

- View details of the work items that you created in Plan for the test. See Viewing details of the work items on page 1637.
- Modify the configuration that you set for Plan on Test. See Managing the DevOps Plan configuration on page 1638.

Viewing details of the work items

After you create a work item in HCL DevOps Plan (Plan) from Test , you can view and track the details of the work item in Plan.

Before you begin

You must have completed the following tasks:

- Completed a test run and the test results are available on the **Results** page.
- Created a work item in Plan against the test results. See Creating work items on page 1632.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Analyze > Results**.

   The **Results** page is displayed and all the test assets that were run earlier are displayed in tabs, as listed in the following table:

Tab	Description
SUITES	Lists all suites, Compound Tests, JMeter tests, JUnit tests, Postman tests, Rate Schedules, Selenium tests, or VU Schedules that are in the selected branch of the repository.
TESTS	Lists all API tests, functional tests, performance tests, or server test assets that are in the selected branch of the repository.
ADVANCED	Lists all assets that are displayed when custom filters are applied for assets in the selected branch of the repository.

Note: You can click the relevant tab to view the tests that were run earlier.

4. Click the test that is listed under the **Test** column.

   The **Results** panel for the test is displayed.

5. Click the **Report** tab in the **Results** panel, if it is not already displayed.
6. Click the work item link under **Defect tracking**.

   The Plan log-in page is displayed.

7. Enter your username and password.
8. Click **Sign In**.

   The work item details are displayed in Plan.

Note: The **Description** field of the work item provides additional details of the test that ran in Test such as the name of the test, the person who ran the test, the verdict of the test result, the date and time of the test run, the links to the test report that is generated in Test, and the link to the query that is created.

Results

You have viewed the details in Plan of the work item that you created in Test.

Managing the DevOps Plan configuration

You can manage the HCL DevOps Plan (Plan) configuration that you set in Test when you want to edit the configuration or when you want to delete the existing configuration.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you obtained the access details and authentication details of Plan from the Plan administrator.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.

3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Click **Add Change Management Integration** option within the **Change Management** section.

5. Perform any of the following actions when you want to edit or delete the Plan configuration:
   - To edit the Plan configuration. Go to Step 6 on page 1639.
   - To delete the Plan configuration. Go to Step 7 on page 1639.
6. Perform the following steps to edit the Plan application that is configured:

   a. Click **Edit** icon .
   The **Modify DevOps Plan configuration** dialog is displayed.

   b. Edit the URL or change it to the one that you want to use in the **URL** field, and then enter the application that you want to use.

   c. Click **Update**.

   The Plan configuration is updated and a confirmation message is displayed.

   d. Go to Results on page 1639.

7. Perform the following steps to delete the existing Plan configuration:

   a. Click the **Delete** icon .

   The **Delete DevOps Plan configuration** dialog is displayed.

   b. Click **Delete**.

   The Plan configuration is deleted and a confirmation message is displayed.

   c. Go to Results on page 1639.

Note: The links to the records that you created for the tests that you ran are not deleted from the results panel in Test even after you delete the Plan configuration.

Results

You achieved the following results:

- Edited and saved the Plan configuration.
- Deleted the Plan configuration.

Integrations with other applications

You can integrate certain applications with Test to run tests, view the test results, and create defects.

Integration plugin compatibility matrix

You can find information about the versions of the integration plugin that are compatible with Test .

The following table lists the versions of the integration plugin that are required to integrate Test with DevOps Deploy, IBM® Engineering Test Management, and Jenkins.

Note: You must download the required version of the integration plugin from My HCLSoftware (MHS) based on the existing version of Test. You can then integrate Test with HCL DevOps Deploy, IBM® Engineering Test Management, and Jenkins.

Test	Engineering Test Management plugin	Jenkins plugin	HCL DevOps Deploy plugin
11.0.9	HCL_DTHub_v1109_ETM_ADAPTER	DTHub-Jenkins-HCL-7.8	DTHub-DD-HCL-5.1 DTHub-DD-HCL-5.1
11.0.8	HCL_DTHub_v1108_ETM_ADAPTER	DTHub-Jenkins-HCL-7.8	DTHub-DD-HCL-5.0 DTHub-DD-HCL-5.0
11.0.7	HCL_DTHub_v1107_ETM_ADAPTER	DTHub-Jenkins-HCL-7.6	DTHub-DD-HCL-5.0 DTHub-DD-HCL-5.0
11.0.6	HCL_DT_HUB_1106_ETM	HCL_DT_HUB_1106_Jenkins	HCL_DT_HUB_1106_Launch HCL_DT_HUB_1106_UCD
11.0.5	HCL_DT_HUB_1105_ETM	HCL_DT_HUB_1105_Jenkins	HCL_DT_HUB_1105_Launch HCL_DT_HUB_1105_UCD
11.0.4	HCL_DT_HUB_1104_ETM	HCL_DT_HUB_1104_Jenkins	HCL_DT_HUB_1104_Launch HCL_DT_HUB_1104_UCD
11.0.3	HCL_DT_HUB_1103_ETM	HCL_DT_HUB_1103_Jenkins	HCL_DT_HUB_1103_Launch HCL_DT_HUB_1103_UCD
11.0.2	HCL_DT_HUB_1102_ETM	HCL_DT_HUB_1102_Jenkins	HCL_DT_HUB_1102_Launch HCL_DT_HUB_1102_UCD

Test	Engineering Test Management plugin	Jenkins plugin	HCL DevOps Deploy plugin
11.0.1	HCL_DT_HUB_1101_ETM	HCL_DT_HUB_1101_Jenkins	HCL_DT_HUB_1101_Launch HCL_DT_HUB_1101_UCD
11.0.0	HCL_DT_HUB_110_ETM	HCL_DT_HUB_110_Jenkins	HCL_DT_HUB_110_Launch HCL_DT_HUB_1101_UCD
10.5.4	HCL_OT_SERVER_1054_ETM	HCL_OT_SERVER_1054_Jenkins	HCL_OT_SERVER_1054_Launch HCL_OT_SERVER_1054_UCD
10.5.3	HCL_OT_SERVER_1053_ETM	HCL_OT_SERVER_1053_Jenkins	HCL_OT_SERVER_1053_Launch HCL_OT_SERVER_1053_UCD
10.5.2	HCL_OT_SERVER_1053_ETM	HCL_OT_SERVER_1052_Jenkins	HCL_OT_SERVER_1052_Launch HCL_OT_SERVER_1052_UCD
10.5.1	HCL_OT_SERVER_1051_ETM	HCL_OT_SERVER_1051_Jenkins	HCL_OT_SERVER_1051_Launch HCL_OT_SERVER_1051_UCD
10.5.0	HCL_OT_SERVER_1050_ETM	HOT-SERVER-Jenkins-6.3	HOT-SERVER-UCD-3.3
10.2.3	HCL_OT_SERVER_1023_ETM	HOT-SERVER-Jenkins-6.1	HOT-SERVER-UCD-3.2
10.2.2	HCL_OT_SERVER_1022_ETM	HOT-SERVER-Jenkins-6.0	HOT-SERVER-UCD-3.0
10.2.1	HCL_OT_SERVER_1021_ETM	HOT-SERVER-Jenkins-5.0	HOT-SERVER-UCD-3.0
10.2.0	NA	HOT-SERVER-Jenkins-5.0	HOT-SERVER-UCD-3.0

Test	Engineering Test Management plugin	Jenkins plugin	HCL DevOps Deploy plugin
10.1.3	NA	HOT-SERVER-Jenkins-4.0	HOT-SERVER-UCD-2.0
10.1.2	NA	HOT-SERVER-Jenkins-4.0	HOT-SERVER-UCD-2.0
10.1.1	NA	HOT-SERVER-Jenkins-3.0	HOT-SERVER-UCD-1.0
10.1.0	NA	HOT-SERVER-Jenkins-2.0	HOT-SERVER-UCD-1.0

Integration with Azure DevOps

When you use Azure DevOps for continuous integration and continuous development of your application, you can run tests created for your application and available in a project on Test , in Azure DevOps pipelines by using the HCL DevOps Test Hub extension.

Overview

You can use the HCL DevOps Test Hub extension to integrate Test with Azure DevOps.

The HCL DevOps Test Hub extension enables you to select any type of test available for your project in Test that you can add to your task for the job in the Azure DevOps pipelines.

You can now follow the tasks listed in the task flow table to integrate Test with Azure DevOps. See Task flow for integrating Azure DevOps on page 1642.

Task flow for integrating DevOps Test with Azure DevOps

The table shows the task flow for integrating Test with Azure DevOps by using the HCL DevOps Test Hub extension.
You must perform these tasks in sequence as listed in the following table. The table also provides you the links to the information about the tasks.

S.N.	Tasks	More information
1	Install Test.	Installation of DevOps Test
2	Create tests in the desktop clients for the application that you want to test.	• DevOps Test Performance documentation • DevOps Test UI documentation • DevOps Test Integrations and APIs documentation

S.N.	Tasks	More information
3	Create a project on Test and add the test assets that are created in the desktop clients to your project.	Managing DevOps Test projects on page 1724
4	Create an organization and a project in Azure DevOps for running jobs in Azure DevOps pipelines.	Creating an organization
5	Access the **Visual Studio Marketplace** portal and search for the latest version of the HCL DevOps Test Hub extension.	Visual Studio Marketplace
6	Install the latest version of the HCL DevOps Test Hub extension.	Installing the HCL DevOps Test Hub extension on page 1643
7	Extend the trust certificate if you have used an internal CA certificate.	Certificate authority: Importing and extending lists on page 732
8	Run tests in an Azure DevOps pipeline.	Running tests in an Azure DevOps Pipeline on page 1644

Related information

Integration with Azure DevOps on page 1642

Installing the HCL DevOps Test Hub extension

You must install the latest version of the HCL DevOps Test Hub extension in your Azure DevOps organization before you can use the extension for running your application tests in an Azure DevOps pipeline.

Before you begin

You must have access to the **Visual Studio Marketplace** portal to install the latest version of the HCL DevOps Test Hub extension.

About this task

After you install the latest version of the HCL DevOps Test Hub extension from the **Visual Studio Marketplace** portal in your Azure DevOps organization, you can select the tests that you want to run for your application in an Azure DevOps pipeline by using the HCL DevOps Test Hub extension.

1. Log in to the **Visual Studio Marketplace** portal, if you are not already logged in.
2. Click the **Azure DevOps** tab.

3. Search for the **HCL DevOps Test Hub extension**.
4. Click **HCL DevOps Test Hub extension**.
5. Click **Get it free**.
   Result

   The **Visual Studio Marketplace** portal for the HCL DevOps Test Hub extension is displayed.
6. Select the organization where you want to run your test from the **Select an Azure DevOps Organization** list.
7. Click **Install**.
   Result

   The installation is completed.
8. Click **Proceed to organization**.
   Result

   The **Organization** page in Azure DevOps is displayed.
9. Click **Organization settings > Extensions**.
   Result

   The HCL DevOps Test Hub extension is displayed as an installed extension.

Results

You have installed the HCL DevOps Test Hub extension in your Azure DevOps organization.

What to do next

You can run tests that are available in Test as a job in an Azure DevOps pipeline. See Running tests in an Azure DevOps Pipeline on page 1644.

Running DevOps Test tests in an Azure DevOps Pipeline

After you install the HCL DevOps Test Hub extension in your organization, you can run tests that are available in Test as a job in Azure DevOps pipelines.

Before you begin

You must have completed the following tasks:

- Added the tests that you created in the desktop clients for your application to the project on Test.
- Installed the latest version of the HCL DevOps Test Hub extension in your organization. See Installing the HCL DevOps Test Hub extension on page 1643.
- Installed an agent in your pipeline. See Azure Pipelines agents.
- Extended the CA list. See Extending the trusted certificate authority list on Linux on page 735 or Extending the trusted certificate authority list on Windows on page 736.
- Generated an offline user token from Test. See Generating an offline token on page 1709.

- Created test cases under test plans if you want to view results of the test runs on the Test Plans dashboard.
- Created encrypted secrets or environment variables for your personal access token (PAT), if you want to use an encrypted PAT.

About this task

After you add the HCL DevOps Test Hub extension in your Azure DevOps organization, you can use an existing pipeline or create a new one to add Test test tasks. You can install an agent or use the one that you installed in your default agent pool. You can add Test tests as tasks to your agent job, configure the task, and then run the task in the Azure DevOps pipeline.

If you have created test cases under test plans in a project on Azure DevOps, you must provide the details of the Azure DevOps URL, test plan, test case, and your PAT while you configure the test job in a pipeline so that you can view the results of the test run on your Test Plan dashboard.

1. Open your **Organization** page in Azure DevOps and perform the following steps:
    a. Click the project you want to use.
    b. Initialize the repository by performing the following steps:
        i. Click **Repos** from the left pane.
        ii. Click **Initialize** from the **Initialize with a README or gitignore** section.

        Note: Select the **Add a README** checkbox if it is not selected.

    c. Click **Pipelines** from the left pane.
    d. Click **Create Pipeline**.
    e. Click **Use the classic editor** to create a pipeline without YAML.
    f. Verify the project, repository, and branch for manual and scheduled builds, and then click **Continue**.
    g. Click **Empty job**.
2. Select **Pipeline** and complete the following steps:
    a. Change the name for the build pipeline if required.
    b. Select the **Agent pool** for your build pipeline.

    You can use the agent from the default agent pool or use the one you have installed.

    c. Select the **Agent Specification** for the agent if required.
3. Add a task to the agent job by completing the following steps:
    a. Click the **Add Task** icon for the agent job.
    Result

The **Add tasks** pane is displayed.

b. Search for the HCL tasks defined in the HCL DevOps Test Hub extension.
Result

The tasks that you can select are displayed.

c. Select the **HCL DevOps Test Hub Task**, and then click **Add** to add the task to the agent job.
Result

The selected task is added to the agent job and it is displayed with a warning that some settings require attention. You must configure the settings mentioned in Step 4 on page 1646.

You can also remove the tasks that are not required in your job. Select the tasks in the list that you want to remove. You can then right-click the tasks, and click **Remove selected task(s)** to remove them.

4. Configure the settings by performing the following steps:

a. Select the task version from the list if required.

b. Follow the action for the task by referring to the following table:

Note: All mandatory fields are marked with an asterisk (*) in the UI.

Field	Description	Action
Display name	The default name of the task. The default name that is displayed is **HCL DevOps Test Hub Task**.	Enter a name for the task that helps you to identify the task by changing the default name.
HCL DevOps Test Hub Task service connection	The details of Test to set up a connection with Azure DevOps.	Select the service connection from the drop-down list. If you are selecting the Test service connection for the first time, you must click **New** to add the following details to add Test service connection, and then save the connection details: ▪ **Server URL** - Enter the URL of Test. The format of the URL is as follows: `https://hostname.` ▪ **Offline Token** - Enter the offline token that you generated from Test. ▪ **Service connection name** - Enter a service connection name.

Field	Description	Action
		▪ **Description (optional)** - Enter the details of the service connection if required. ▪ Optionally, select the **Grant access permission to all pipelines** checkbox if required. You can save the service connection details. The service connection is available for selection from the Test service connection drop-down list. **Note:** You can edit or delete the service connection that you added if required. Click **Manage**, and then select the service connection from the **Service connections** list to open the service connection. You can click **Edit** to edit the service connection details. If you want to delete an existing service connection details, click the **Vertical ellipsis** icon, and then click **Delete** to delete the service connection details. Optionally, you can also create, edit, and delete a service connection from your Azure DevOps project dashboard. Open your project, click **Project settings > Service connections**, and then perform any of the following tasks: ▪ Create a new service connection. ▪ Edit or delete an existing service connection that you already added.
Team Space Name	The name of the team space on Test that contains your project with the test assets.	Enter the name of the team space that contains the project. **Note:** The license on the team space must be configured and you must be a member of that team space.
Project	The name of the project containing the test assets in your team space on Test.	Enter the name of the project from the available names in the team space. **Note:** **Note:** The **Project** drop-down list displays the projects where you are an *Owner* or a *member* of Test. You must be an *Owner* or a *Tester* of the

Field	Description	Action
		project that is available in the team spaces to run the tests from the Jenkins server. Azure DevOps pipeline.
Branch	The name of the branch of the repository in your project on Test that contains the test assets.	Enter the branch where the test assets are stored.
Repository Link	The URL or link to the repository in your project that contains the test assets.	Enter the repository path that is configured for the project that contains the test to run.
File Path	The path of the test in the repository of your project.	Enter the path of the test in Test that you want to run.
API Test Environment	The environment defined in the API Suite or test to use in the test run.	Enter the name of the environment that is configured in the test asset for the test created in Test Integrations and APIs. **Note:** This field is applicable only if you are running an API suite test.
Variables	The variable name and its value.	Enter the name of the variable and its value if your test requires variables during the test run time. You must enter the variables in the following format: `name_of_the_variable=value_of_the_varibale` You can add multiple variables to the test run separated by a semicolon. For example, *varname1=value1;varname2=value2*
Datasets	The name with path to the dataset that must be used during the test run.	Enter the path to the dataset if you want to replace the values of the dataset during a test run. You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format:

Field	Description	Action
		`/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv` **Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.
Labels	The option to add labels to the test results when the test run is complete.	Enter the labels to add to test results when the test run is complete. You can add multiple labels to a test result separated by a comma. For example, *label1, label2* After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of Test.
Classification Name	The name of the classification that contains the secrets to be used in certain API Suites or tests.	Enter the name of the classification that contains the secrets to be used in the API Suite or test. **Note:** This field is applicable only if you are running an API suite test.
Azure DevOps Project URL	The URL of the test project in the organization on the Azure DevOps server. **Note:** You must enter the details for this option if you have created test cases under test plans in your Azure project and want to view the test	Enter the URL of the test project in the organization on the Azure DevOps server in the following format: https://<host>/<orgname>/<projectName> You must use this option with the following options: ▪ Azure DevOps PAT ▪ Test Plan Name ▪ Test Case Name

Field	Description	Action
	results on the Test Plans Dashboard.	
Azure DevOps PAT	The personal access token (PAT) of the Azure server where the test project that contains the test plan is hosted. You can also enter the variable for the encrypted PAT that you create in Azure DevOps. **Note:** The token must have read and write access.	Enter your personal access token. You must use this option with the following options: ▪ Azure DevOps Project URL ▪ Test Plan Name ▪ Test Case Name
Test Plan Name	The name of the test plan in your Azure project.	Enter the name of the test plan. You must use this option with the following options: ▪ Azure DevOps Project URL ▪ Azure DevOps PAT ▪ Test Case Name
Test Case Name	The name of the test case under the test plan in your Azure project.	Enter the name of the test case. You must use this option with the following options: ▪ Azure DevOps Project URL ▪ Azure DevOps PAT ▪ Test Plan Name
Fail the job if the test or any of the containing tests fail	The flag to indicate whether the job should be marked as failed when the test or any child test within the test fails.	Select the checkbox.

Field	Description	Action
Export JUnit Report	The location to store the JUnit report of the test.	Enter the path to store the JUnit report that is generated for the test in the XML format. You must specify the complete path to the directory including the file name of the report. For example, `C:/Users/Documents/Reports/junit_test_report.xml`

c. Expand **Control Options** and configure the settings for your task if required.

d. Expand **Output Variables** and configure the settings for your task if required.

5. Select from the following options:

a. Click **Save** to save the configured settings for the task.

Note: The task is not queued for a run.

You can save the task to a build pipeline and opt to run the build at a later time.

b. Click **Save & queue** to save the configurations and queue the run in the pipeline.
Result

The **Run pipeline** dialog is displayed.

6. Complete the following steps:

a. Enter a comment for the test in the **Save comment** field.

b. Select the agent that you configured for the test from the **Agent pool** list.

c. Select the agent specification from the **Agent Specification** list if required.

d. Select the branch from the **Branch/tag** list.

e. Add the variables and demands for the task run from the **Advanced Options** pane if required.

f. Select the **Enable system diagnostics** checkbox for a detailed log view.

g. Click **Save and run**.
Result

The `pipeline summary` page displays the progress of the job run.

Results

You have run the tests for the application that you are testing in the Azure DevOps pipeline.

What to do next

You can perform any of the following actions in Azure DevOps:

- View the test results on the **Task** page by clicking the task.

  The **Reports information** section on the **Task** page displays the names of the report along with its corresponding URLs. The report URLs are the Test URLs where the reports are stored. You can access the report URLs to view the test run information at any point of time.
- View the task logs from the **pipeline summary** page for the job you ran.
- View the test results on the Test Plans Dashboard if you created test cases under test plans in your Azure project.

You can perform any of the following actions in Test:

- View the test results of the test that you ran from Azure DevOps on the **Results** page.
- View the logs and reports of the test that you ran from Azure DevOps by clicking the result panel of the test asset or resource, and then click the log or the report you want to view.

Integration with HCL DevOps Deploy

You can integrate Test with HCL DevOps Deploy (Deploy) to run test assets as a **Process** from Deploy.

When you use Deploy for automating the deployments of your applications to an environment during the development cycle, you can create test assets for your application either in Test or desktop clients. After you configure the project in Test, you can then run those tests assets from Deploy.

You must use the Test Deploy plugin to integrate Test with Deploy. Integrating Test with Deploy automates the process of test runs. If you have many tests to run at regular intervals, you can use Deploy to initiate test runs automatically.

The following table lists the tasks that you must perform to run performance test assets from Deploy:

Tasks	Go to...
Create any or all the following types of test assets in desktop clients to test your application: • API test assets • Functional test assets • Performance test assets	• DevOps Test Performance documentation • DevOps Test UI documentation • DevOps Test Integrations and APIs documentation
Create a project, and then configure the repository in Test.	Working with projects in a team space on page 1725
Install Deploy with at least one agent.	HCL® Launch Documentation.

Tasks	Go to...
	System Requirements for information about specific versions of Deploy requirements.
Install the DevOps Test Hub Deploy plugin on Deploy.	Installing the plugin on HCL DevOps Deploy on page 1653.
Create a **Component** in Deploy.	Creating a component in HCL DevOps Deploy on page 1655.
Create a **Process** for the component in Deploy.	Creating a process in HCL DevOps Deploy on page 1656.
Configure the **Process** that you created for the component in Deploy.	Configuring the process in HCL DevOps Deploy on page 1657.
Create a **Resource** in Deploy.	Creating a resource in HCL DevOps Deploy on page 1661.
Configure the **Resource** in Deploy.	Configuring the resource in HCL DevOps Deploy on page 1662.
Create an **Application** in Deploy.	Creating an application in HCL DevOps Deploy on page 1663.
Configure the **Application** that you use to run performance test assets in Deploy.	Configuring the application HCL DevOps Deploy on page 1664.
Run the test assets from Deploy.	Running tests from HCL DevOps Deploy on page 1666.

Installing the plugin on HCL DevOps Deploy

You must install the DevOps Test Hub Deploy plugin to run test assets from HCL DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Installed Deploy. For more information about the installation, refer to the HCL DevOps Deploy documentation.
  - Been granted access to Deploy.
  - Downloaded the DevOps Test Hub Deploy plugin from the My HCLSoftware (MHS).

    For more information about specific versions of plug-in, see Integration plugin compatibility matrix on page 1639.

1. Log in to Deploy, if you are not already logged in.

   Result

   The Deploy dashboard is displayed.
2. Click **Settings**.

3. Click **Installed Plugins** from the **Automation** pane.
   If you are using Deploy 8.1.1 or older version, then you must click **Automation Plugins** from the **Automation** pane.
4. Click **Load plugin**.
5. Click **Choose File** to locate and open the compressed DevOps Test Hub Deploy plugin file.

   Remember: You must not extract the DevOps Test Hub Deploy plugin compressed file contents.

6. Click **Submit**.
   Result

   The installed plugin is displayed on the **Automation Plugins** tab.

Results

You have installed the DevOps Test Hub Deploy plugin on Deploy.

What to do next

You must create a **Component** in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.

Related information

Installing plug-ins from the Web UI of Deploy

Using a custom trust store for HCL DevOps Deploy

You can use a custom trust store in the DevOps Test Hub Deploy plugin file to establish a trusted and secure connection between the HCL DevOps Deploy (Deploy) server and Test .

Before you begin

You must have completed the following tasks:

You must have completed the following tasks:

- Saved the Certificate Authority (CA) assigned to Test.

  Note: You can contact the administrator of Test if you do not have a copy of the CA.

- Installed Java on the computer from where you access the Test UI.

About this task

If the SSL certificate assigned to Test is signed by an internal Certified Authority (CA), then you must download and import the CA certificate to a custom trust store. You can then use the custom trust store in the Test test process step to establish a trusted and secure connection between the Deploy server and Test.

Note: If the internal CA certificate is already imported to the default trust store that is used by the Deploy server, you need not use a custom trust store.

1. Locate the default trust store file (`cacerts` file) from the JRE directory.
   For example: `C:\Program Files\Java\jdk-18.0.1.1\lib\security`
2. Copy the `cacerts` file to a location of your choice on your computer.
3. Place the CA assigned to Test in the same location where you saved the `cacerts` file.
4. Run the following command from the command-line interface or terminal to import the CA of Test to custom trust store:

   ```
   keytool -import -trustcacerts -file {path to the downloaded CA with the file extension} -alias {label
    for the certificate} -keystore {path to the trust store cacerts file}
   ```

   For example,

   ```
   keytool -import -trustcacerts -file C:\Users\Desktop\cert\ca-master-hcl.crt -alias "otsca" -keystore C:
   \Users\Desktop\cert\cacerts
   ```

   Note: The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore {path to the trust store cacerts file}
   ```

   For example, `keytool -storepasswd -keystore C:\Users\Desktop\cert\cacerts`

Results

You have successfully imported the downloaded CA to the custom trust store.

What to do next

You can then run the tests that are available in your Test project from the Deploy server.

Creating a component in HCL DevOps Deploy

You must create a component to include artifacts and processes. The artifacts include runnable files, images, databases, configuration instructions. Whereas the processes define the activities that components can perform.

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Components**, and then click **Create Component**.

3. Enter a name for the component in the **Name** field.
4. Enter the details in the other optional fields based on your requirement, and then click **Save**.

   Result

   The component that you created is displayed.

Results

You have created the component in Deploy.

What to do next

You must create a process for the component in Deploy. See Creating a process in HCL DevOps Deploy on page 1656.

Related information

HCL DevOps Deploy Documentation

Creating a process in HCL DevOps Deploy

You must create a process for the component to include step properties for the test that you want to run from HCL DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.

1. Log in to Deploy, if you are not already logged in.

   Result

   The Deploy dashboard is displayed.
2. Click **Components**.

   Result

   A list of components that are available in Deploy is displayed.
3. Select the component from the list for which you want to create a process.
4. Click the **Processes** tab, and then click **Create Process**.

   Result

   The **Create Process** dialog is displayed.
5. Enter a name for the process in the **Name** field.
6. Select **Operational (No Version Needed)** from **Process Type** drop-down list.
7. Verify the **Default Working Directory** field.

The **Default Working Directory** field defines the location that the agent uses to run the process. The default value is `${p:resource/work.dir}/${p:component.name}`.

Where `${p:resource/work.dir}` is the default working directory for the agent and `${p:component.name}` is the name of the component.

8. Click **Save**.
   Result
   The process that you created is listed in the **Processes** tab and the **Design** tab for the process is displayed.

Results

You have created the process for the component in Deploy.

What to do next

You must configure the process in Deploy. See Configuring the process in HCL DevOps Deploy on page 1657.

Related information

HCL DevOps Deploy Documentation

Configuring the process in HCL DevOps Deploy

You must configure the process that you created for the component to organize the steps in the process, specify the properties of the steps, and connect them.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.
  - Created a process for the component in Deploy. See Creating a process in HCL DevOps Deploy on page 1656.

About this task

When you open any process to configure, the process is displayed in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and steps are automatically placed on the design area.

You must provide the values for certain fields in the properties for the selected test step to run tests from Deploy.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Components**.
   Result
   A list of components that are available in Deploy is displayed.
3. Select the component from the list in which you created the process.
4. Click the **Processes** tab.
   Result
   A list of processes that are available for the component is displayed.
5. Select the process from the list that you want to configure.
   Result
   The **Design** tab for the process is displayed.
6. Click **HCL DevOps Test Workbench**, and then **HCL DevOps Test Hub** from the left menu.
7. Drag the **Run HCL DevOps Test Hub test** step, and then drop it into the design area.

   Note: The selected test must be placed between **Start** and **Finish** steps.

8. Specify the properties for the selected test by performing the following steps:

   a. Click the **Edit** icon.
      Result
      The **Edit Properties for Run HCL DevOps Test Hub test** dialog is displayed.

   b. Specify the properties for the selected test step by referring to the following table:

      The following table lists the required fields that you must provide to run the test from Deploy:

Fields	Action
Name	Enter the name for the test step.
HCL DevOps Test Hub URL	Enter the URL of Test. If you do not have the details of the custom trust store, you must append #insecure-skip-tls-verify to the URL. For example, https://10.134-.56.236.nip.io/test#insecure-skip-tls-verify
Offline Token	Enter the offline token that you generated from Test.
Team Space Name	Enter the name of the team space that contains the project.

Fields	Action
	Note: The license for the team space must be configured and you must be a member of that team space.
Project Name	Enter the name of the project that are available in the team space.
Repository Link	Enter the URL of the Git repository that you added to your project in Test.
File Path	Enter the path of the test assets that you want to run. You can find the path of the test assets from the **Execution** page in Test

The following table lists the optional fields that you can provide to run the test from Deploy:

Fields	Description
Branch Name	Use this field to enter the name of the branch in the Git repository where you stored test assets.
Datasets	Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run. You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format: `/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv` **Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.
Environment	Use this field to enter the environment details for the following types of test assets: ▪ APISUITE ▪ APITEST ▪ APISTUB
Labels	Use this field to add labels to test results when the test run is complete.

Fields	Description
	You can add multiple labels to a test result separated by a comma. For example, *label1, label2* After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of Test.
Classification Name	Use this field to enter the name of the Classification if you created secrets for your project and to use Secrets at test run time. This field is mandatory only if you want to run the following types of test assets: ▪ APISUITE ▪ APITEST ▪ APISTUB
Show Hidden Properties	Select this checkbox to display the additional fields that are hidden. You can configure the following fields if you have the details of a custom trust store: ▪ **Custom Trust Store Password**: Use this field to enter the trust store password if you have modified the password while creating the custom trust store. **Note:** If you have not modified the trust store password, you can retain the **Custom Trust Store Password** field blank. ▪ **Custom Trust Store Path**: Use this field to enter the file path of your trust store followed by the file name if Test uses an internal CA certificate and you have imported the certificate to a custom trust store. Also, you can configure the HTTP Proxy to connect to Test by using the following fields: ▪ **HTTP Proxy Host**: Use this field to enter the hostname of the HTTP proxy that you want to connect to Test. ▪ **HTTP Proxy Port**: Use this field to enter the port number where the HTTP proxy is used to listen to both HTTP and HTTPS traffic. ▪ **HTTP Proxy User name**: Use this field to provide the username for the HTTP proxy. ▪ **HTTP Proxy Password**: Use this field to enter a valid password for the user that you specified in the **HTTP Proxy User name** field. **Note:** You can use the HTTP-related fields only when you want to run the following types of test assets:

Fields	Description
	▪ APISUITE ▪ APITEST ▪ APISTUB
Variables	Use this field to enter the name of the variable and its value if your test requires variables during the test run time. You must enter the variables in the following format: `name_of_the_variable=value_of_the_varibale` You can add multiple variables to the test run separated by a semicolon. For example, *varname1=value1;varname2=value2*

Note: In addition to optional fields, you can also use the following fields from Deploy to configure your test run:

- **Working Directory**
- **Precondition**
- **Post Processing Script**
- **Use Impersonation**
- **Auth Token Restriction**

You can accept the default values or change the values based on your requirements. For more information about these fields, see the related links.

c. Click **OK** to save the properties for the test.

9. Click **Save** in the design area.

Results

You have configured the process for the component in Deploy.

What to do next

You must create a resource in Deploy. See Creating a resource in HCL DevOps Deploy on page 1661.

Creating a resource in HCL DevOps Deploy

You must create a resource to associate agents with components that you created in HCL DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Resources**, and then click **Create Top-Level Group**.
   Result
   The **Create Resource** dialog is displayed.
3. Enter a name for the resource in the **Name** field.
4. Click **Save**.

Results

You have created the resource in Deploy.

What to do next

You must configure the resource. See Configuring the resource in HCL DevOps Deploy on page 1662.

Configuring the resource in HCL DevOps Deploy

You must configure the resource to add an agent and associate the agent with the component.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.
  - Created a resource in Deploy. See Creating a resource in HCL DevOps Deploy on page 1661.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Resources**.
   Result
   A list of resources that are available in Deploy is displayed.
3. Perform the following steps to add an agent to the resource:
   a. Click the resource from the list for which you want to add an agent.
   b. Click the **Actions** icon from the last column, and then click **Add Agent**.

c. Select the agent from the drop-down list.

Note: The **Name** field is auto populated with the name of the agent.

d. Click **Save**.

Result

The selected agent is added to the resource and you can view the status of the agent in the **Status** column.

4. Perform the following steps to add a component to the agent:

a. Click the agent from the list for which you want to add a component.

b. Click the **Actions** icon from the last column, and then click **Add Component**.

c. Select the component from the drop-down list.

Note: The **Name** field is auto populated with the name of the component.

d. Click **Save**.

Result

The selected component is added to the agent.

Results

You have configured the resource in Deploy.

What to do next

You must create an application. See Creating an application in HCL DevOps Deploy on page 1663.

Creating an application in HCL DevOps Deploy

You must create an application to fetch all the components together that you want to deploy.

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.

Result

The Deploy dashboard is displayed.

2. Click **Applications**.
3. Click **Create Applications**, and then **New Applications**.
4. Enter a name for the application in the **Name** field.

Result

The **Environments** page for the application that you created is displayed.

Results

You have created the application in Deploy.

What to do next

You must configure the application. See Configuring the application HCL DevOps Deploy on page 1664.

Configuring the application HCL DevOps Deploy

You must configure the application to associate resources with environments and define processes to run test assets.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Created a process for the component in Deploy. See Creating a process in HCL DevOps Deploy on page 1656.
  - Created a component in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.
  - Created a process for the component in Deploy. See Creating a process in HCL DevOps Deploy on page 1656.
  - Configure the process for the component in Deploy. See Configuring the process in HCL DevOps Deploy on page 1657.
  - Created a resource in Deploy. See Creating a resource in HCL DevOps Deploy on page 1661.
  - Configured the resource in Deploy. See Configuring the resource in HCL DevOps Deploy on page 1662.
  - Created an application in Deploy. See Creating an application in HCL DevOps Deploy on page 1663.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Applications**.
   Result
   A list of applications that are available in Deploy is displayed.
3. Click the application that you want to configure from the **Name** column.
   Result
   The **Environments** page for the selected application is displayed.
4. Perform the following steps to create an environment for the application that you selected:

a. Click **Create Environment**.
b. Enter a name for the environment in the **Name** field.
c. Click **Save**.

5. Perform the following steps to configure resources to the environment:

   a. Click the environment that you created.

   b. Click **Add Base Resources**.
   Result
   A list of resources that are available in Deploy is displayed.

   c. Select the checkbox to add resources to the environment.

   d. Click **Save**.
   Result
   You can view the corresponding agent and the component that you added for the resource by using the **Expand** icon.

6. Perform the following steps to add the component to the application:
   a. Click **Applications**, and then select your application from the list.
   b. Click the **Components** tab, and then **Add Components**.
   c. Select the checkbox from the drop-down list to add components to the application.
   d. Click **Save**.
7. Perform the following steps to create a process for the application:

   a. Click the **Processes** tab, and then **Create Process**.

   b. Enter a name for the process in the **Name** field.

   c. Click **Save**.
   Result
   The **Design** tab for the process that you created is displayed.

8. Drag the component process listed under the **Component Process Steps** option from the left navigation pane and drop it into the design area.
9. Select the component process from the drop-down list in the **Operational (No Version Needed) Process** field.
10. Click **Save**.
11. Click the **Edit** icon, and then change the name of the properties.
12. Click **OK**, and then click **Save**.

Results

You have configured the application to run test assets from Deploy.

What to do next

You can run test assets from Deploy. See Running tests from HCL DevOps Deploy on page 1666.

Related information

HCL DevOps Deploy Documentation

Running tests from HCL DevOps Deploy

You can run test assets from HCL DevOps Deploy (Deploy) as a **Process** to test your application under test.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in HCL DevOps Deploy on page 1655.
  - Created a process for the component in Deploy. See Creating a process in HCL DevOps Deploy on page 1656.
  - Configure the process for the component in Deploy. See Configuring the process in HCL DevOps Deploy on page 1657.
  - Created a resource in Deploy. See Creating a resource in HCL DevOps Deploy on page 1661.
  - Configured the resource in Deploy. See Configuring the resource in HCL DevOps Deploy on page 1662.
  - Created an application in Deploy. See Creating an application in HCL DevOps Deploy on page 1663.
  - Configured the application in Deploy. See Configuring the application HCL DevOps Deploy on page 1664.
  - Started the Deploy agent. For more information, refer to the HCL Launch documentation.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Applications**.
   Result
   A list of applications that are available in Deploy is displayed.
3. Click the application that you configured for the test run.
4. Click the **Request Process** icon.
   Result
   The **Create Deployment** page is displayed.
5. Select the process that you created for the application which contains the test assets from the **Process** drop-down list.
6. Click **Next**, and then select the specific version of component, if any.

Note: If you have multiple versions for your component, then only you can choose the required version by clicking the **Add** option.

7. Click **Next**, and then verify the properties for the test run.
8. Click **Next**, and then verify that the status of the **Run Now** option is enabled.
   You can disable the **Run Now** option, and then use the **Date**, **Time**, and **Recurrence Pattern** fields to run tests as **Process** at the scheduled time.
9. Click **Next**, and then verify the details about the **Process**.
10. Click **Submit Deployment** to initiate the test run.

Results

You have run the performance test assets as **Process** from Deploy.

What to do next

You can view the status of the process as follows:

- **Success**: When the test run is successful
- **Failed**: When the test run fails

You can view the details of the test run in the **Execution Log** section. The **Reports information** section on the Log file displays the names of the report along with its corresponding URLs.

Integration with IBM® DevOps Deploy

You can integrate Test with IBM® DevOps Deploy (Deploy) to run test assets as a **Process** from Deploy.

When you use Deploy for automating the deployments of your applications to an environment during the development cycle, you can create test assets for your application in desktop clients. After you configure the project in Test, you can then run those tests assets from Deploy.

You must use the DevOps Test Hub Deploy plugin to integrate Test with Deploy. Integrating Test with Deploy automates the process of test runs. If you have many tests to run at regular intervals, you can use Deploy to initiate test runs automatically.

The following table lists the tasks that you must perform to run performance test assets from Deploy:

Tasks	Go to...
Create any or all the following types of test assets in desktop clients to test your application: • API test assets • Functional test assets • Performance test assets	• DevOps Test Performance documentation • DevOps Test UI documentation • DevOps Test Integrations and APIs documentation

Tasks	Go to...
Create a project, and then configure the repository in Test.	Working with projects in a team space on page 1725
Install Deploy with at least one agent.	Installing the server System Requirements for information about specific versions of Deploy requirements.
Install the DevOps Test Hub Deploy plugin on Deploy.	Installing the DevOps Test Hub Deploy plugin on page 1668.
Create a **Component** in Deploy.	Creating a component in IBM DevOps Deploy on page 1670.
Create a **Process** for the component in Deploy.	Creating a process in IBM DevOps Deploy on page 1671.
Configure the **Process** that you created for the component in Deploy.	Configuring the process in IBM DevOps Deploy on page 1672.
Create a **Resource** in Deploy.	Creating a resource in IBM DevOps Deploy on page 1676.
Configure the **Resource**.	Configuring the resource in IBM DevOps Deploy on page 1677.
Create an **Application** in Deploy.	Creating an application in IBM DevOps Deploy on page 1678.
Configure the **Application** that you want to use to run test assets from Deploy.	Configuring the application in IBM DevOps Deploy on page 1679.
Run the test assets from Deploy.	Running tests from IBM DevOps Deploy on page 1681.

Installing the DevOps Test Hub Deploy plugin

You must install the DevOps Test Hub Deploy plugin to run test assets from IBM® DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Installed Deploy. For more information about the installation, refer to the IBM Urban Code documentation.
  - Been granted access to Deploy.
  - Downloaded the DevOps Test Hub Deploy plugin from the My HCLSoftware (MHS).

    For more information about specific versions of plug-in, see Integration plugin compatibility matrix on page 1639.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Settings**.
3. Click **Installed Plugins** from the **Automation** pane.
   If you are using Deploy 8.1.1 or older version, then you must click **Automation Plugins** from the **Automation** pane.
4. Click **Load plugin**.
5. Click **Choose File** to locate and open the compressed DevOps Test Hub Deploy plugin file.

   Remember: You must not extract the DevOps Test Hub Deploy plugin compressed file contents.

6. Click **Submit**.
   Result

   The installed plugin is displayed on the **Automation Plugins** tab.

Results

You have installed the DevOps Test Hub Deploy plugin on Deploy.

What to do next

You must create a **Component** in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.

Related information

Installing plug-ins from the Web UI of Deploy

Using a custom trust store for IBM® DevOps Deploy

You can use a custom trust store in the DevOps Test Hub Deploy plugin file to establish a trusted and secure connection between the IBM® DevOps Deploy (Deploy) server and Test .

Before you begin

You must have completed the following tasks:

- Saved the Certificate Authority (CA) assigned to Test.

  Note: You can contact the administrator of Test if you do not have a copy of the CA.

- Installed Java on the computer from where you access the Test UI.

About this task

If the SSL certificate assigned to Test is signed by an internal Certified Authority (CA), then you must download and import the CA certificate to a custom trust store. You can then use the custom trust store in the **Run HCL DevOps Test Hub test** process step to establish a trusted and secure connection between the Deploy server and Test.

Note: If the internal CA certificate is already imported to the default trust store that is used by the Deploy server, you need not use a custom trust store.

1. Locate the default trust store file (`cacerts` file) from the JRE directory.
   For example: `C:\Program Files\Java\jdk-18.0.1.1\lib\security`
2. Copy the `cacerts` file to a location of your choice on your computer.
3. Place the CA assigned to Test in the same location where you saved the `cacerts` file.
4. Run the following command from the command-line interface or terminal to import the CA of Test to custom trust store:

   ```
   keytool -import -trustcacerts -file {path to the downloaded CA with the file extension} -alias {label
    for the certificate} -keystore {path to the trust store cacerts file}
   ```

   For example,

   ```
   keytool -import -trustcacerts -file C:\Users\Desktop\cert\ca-master-hcl.crt -alias "otsca" -keystore C:
   \Users\Desktop\cert\cacerts
   ```

   Note: The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore {path to the trust store cacerts file}
   ```

   For example, `keytool -storepasswd -keystore C:\Users\Desktop\cert\cacerts`

Results

You have successfully imported the downloaded CA to the custom trust store.

What to do next

You can then run the tests that are available in your Test project from the Deploy server.

Creating a component in IBM® DevOps Deploy

You must create a component to include artifacts and processes in IBM® DevOps Deploy (Deploy). The artifacts include runnable files, images, databases, configuration instructions. Whereas the processes define the activities that components can perform.

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Components**, and then click **Create Component**.
3. Enter a name for the component in the **Name** field.
4. Enter the details in the other optional fields based on your requirement, and then click **Save**.
   Result
   The component that you created is displayed.

Results

You have created the component in Deploy.

What to do next

You must create a process for the component in Deploy. See Creating a process in IBM DevOps Deploy on page 1671.

Creating a process in IBM® DevOps Deploy

You must create a process for the component to include step properties for the test that you want to run from IBM® DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Components**.
   Result
   A list of components that are available in Deploy is displayed.
3. Select the component from the list for which you want to create a process.
4. Click the **Processes** tab, and then click **Create Process**.
   Result
   The **Create Process** dialog is displayed.
5. Enter a name for the process in the **Name** field.
6. Select **Operational (No Version Needed)** from **Process Type** drop-down list.
7. Verify the **Default Working Directory** field.

The **Default Working Directory** field defines the location that the agent uses to run the process. The default value is `${p:resource/work.dir}/${p:component.name}`.

Where `${p:resource/work.dir}` is the default working directory for the agent and `${p:component.name}` is the name of the component.

8. Click **Save**.
   Result
   The process that you created is listed in the **Processes** tab and the **Design** tab for the process is displayed.

Results

You have created the process for the component in Deploy.

What to do next

You must configure the process in Deploy. See Configuring the process in IBM DevOps Deploy on page 1672.

Configuring the process in IBM® DevOps Deploy

You must configure the process that you created for the component to organize the steps in the process, specify the properties of the steps, and connect them in IBM® DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.
  - Created a process for the component in Deploy. See Creating a process in IBM DevOps Deploy on page 1671.
  - Generated an offline user token from Test. See Generating an offline token on page 1709.
  - Added the tests assets to the project on Test.

About this task

When you open any process to configure, the process is displayed in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and steps are automatically placed on the design area.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Components**.

Result

A list of components that are available in Deploy is displayed.

3. Select the component from the list in which you created the process.
4. Click the **Processes** tab.

   Result

   A list of processes that are available for the component is displayed.
5. Select the process from the list that you want to configure.

   Result

   The **Design** tab for the process is displayed.
6. Click **HCL DevOps Test Workbench**, and then **HCL DevOps Test Hub** from the left menu.
7. Drag the **Run HCL DevOps Test Hub test** step, and then drop it into the design area.

   Note: The selected test must be placed between **Start** and **Finish** steps.

8. Specify the properties for the selected test by performing the following steps:

   a. Click the **Edit** icon.

      Result

      The **Edit Properties for Run HCL DevOps Test Hub test** dialog is displayed.

   b. Specify the properties for the selected test step by referring to the following tables:

      The following table lists the required fields that you must provide to run the test from Deploy:

Fields	Action
Name	Enter the name for the test step.
HCL DevOps Test Hub URL	Enter the URL of Test. If you do not have the details of the custom trust store, you must append #insecure-skip-tls-verify to the URL. For example, https://10.134-.56.236.nip.io/test#insecure-skip-tls-verify
Offline Token	Enter the offline token that you generated from Test.
Team Space Name	Enter the name of the team space that contains the project.

Fields	Action
	Note: The license for the team space must be configured and you must be a member of that team space.
Project Name	Enter the name of the project that are available in the team space.
Repository Link	Enter the URL of the Git repository that you added to your project in Test.
File Path	Enter the path of the test assets that you want to run. You can find the path of the test assets from the **Execution** page in Test

The following table lists the optional fields that you can provide to run the test from Deploy:

Fields	Description
Branch Name	Use this field to enter the name of the branch in the Git repository where you stored test assets.
Datasets	Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run. You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format: `/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv` **Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.
Environment	Use this field to enter the environment details for the following types of test assets: ▪ APISUITE ▪ APITEST ▪ APISTUB
Labels	Use this field to add labels to test results when the test run is complete.

Fields	Description
	You can add multiple labels to a test result separated by a comma. For example, *label1, label2* After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of Test.
Classification Name	Use this field to enter the name of the Classification if you created secrets for your project and to use Secrets at test run time. This field is mandatory only if you want to run the following types of test assets: ▪ APISUITE ▪ APITEST ▪ APISTUB
Show Hidden Properties	Select this checkbox to display the additional fields that are hidden. You can configure the following fields if you have the details of a custom trust store: ▪ **Custom Trust Store Password**: Use this field to enter the trust store password if you have modified the password while creating the custom trust store. **Note:** If you have not modified the trust store password, you can retain the **Custom Trust Store Password** field blank. ▪ **Custom Trust Store Path**: Use this field to enter the file path of your trust store followed by the file name if Test uses an internal CA certificate and you have imported the certificate to a custom trust store. Also, you can configure the HTTP Proxy to connect to Test by using the following fields: ▪ **HTTP Proxy Host**: Use this field to enter the hostname of the HTTP proxy that you want to connect to Test. ▪ **HTTP Proxy Port**: Use this field to enter the port number where the HTTP proxy is used to listen to both HTTP and HTTPS traffic. ▪ **HTTP Proxy User name**: Use this field to provide the username for the HTTP proxy. ▪ **HTTP Proxy Password**: Use this field to enter a valid password for the user that you specified in the **HTTP Proxy User name** field. **Note:** You can use the HTTP-related fields only when you want to run the following types of test assets:

Fields	Description
	▪ APISUITE ▪ APITEST ▪ APISTUB
Variables	Use this field to enter the name of the variable and its value if your test requires variables during the test run time. You must enter the variables in the following format: `name_of_the_variable=value_of_the_varibale` You can add multiple variables to the test run separated by a semicolon. For example, *varname1=value1;varname2=value2*

Note: In addition to optional fields, you can also use the following fields from Deploy to configure your test run:

- **Working Directory**
- **Precondition**
- **Post Processing Script**
- **Use Impersonation**
- **Auth Token Restriction**

You can accept the default values or change the values based on your requirements. For more information about these fields, see the related links.

c. Click **OK** to save the properties for the test.

9. Click **Save** in the design area.

Results

You have configured the process for the component in Deploy.

What to do next

You must create a resource in Deploy. See Creating a resource in IBM DevOps Deploy on page 1676.

Creating a resource in IBM® DevOps Deploy

You must create a resource to associate agents with components that you created in IBM® DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Resources**, and then click **Create Top-Level Group**.
   Result
   The **Create Resource** dialog is displayed.
3. Enter a name for the resource in the **Name** field.
4. Click **Save**.

Results

You have created the resource in Deploy.

What to do next

You must configure the resource. See Configuring the resource in IBM DevOps Deploy on page 1677.

Configuring the resource in IBM® DevOps Deploy

You must configure the resource to add an agent and associate the agent with the component in IBM® DevOps Deploy (Deploy).

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.
  - Created a resource in Deploy. See Creating a resource in IBM DevOps Deploy on page 1676.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Resources**.
   Result
   A list of resources that are available in Deploy is displayed.
3. Perform the following steps to add an agent to the resource:
   a. Click the resource from the list for which you want to add an agent.
   b. Click the **Actions** icon from the last column, and then click **Add Agent**.

c. Select the agent from the drop-down list.

Note: The **Name** field is auto populated with the name of the agent.

d. Click **Save**.

Result

The selected agent is added to the resource and you can view the status of the agent in the **Status** column.

4. Perform the following steps to add a component to the agent:

a. Click the agent from the list for which you want to add a component.

b. Click the **Actions** icon from the last column, and then click **Add Component**.

c. Select the component from the drop-down list.

Note: The **Name** field is auto populated with the name of the component.

d. Click **Save**.

Result

The selected component is added to the agent.

Results

You have configured the resource in Deploy.

What to do next

You must create an application. See Creating an application in IBM DevOps Deploy on page 1678.

Creating an application in IBM® DevOps Deploy

You must create an application in IBM® DevOps Deploy (Deploy) to fetch all the components together that you want to deploy.

Before you begin

- You must be familiar with working with Deploy.
- You must have been granted access to Deploy.

1. Log in to Deploy, if you are not already logged in.

Result

The Deploy dashboard is displayed.

2. Click **Applications**.
3. Click **Create Applications**, and then **New Applications**.
4. Enter a name for the application in the **Name** field.

Result

The **Environments** page for the application that you created is displayed.

Results

You have created the application in Deploy.

What to do next

You must configure the application. See Configuring the application in IBM DevOps Deploy on page 1679.

Configuring the application in IBM® DevOps Deploy

You must configure the application in IBM® DevOps Deploy (Deploy) to associate resources with environments and define processes to run test assets.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.
  - Created a process for the component in Deploy. See Creating a process in IBM DevOps Deploy on page 1671.
  - Configure the process for the component in Deploy. See Configuring the process in IBM DevOps Deploy on page 1672.
  - Created a resource in Deploy. See Creating a resource in IBM DevOps Deploy on page 1676.
  - Configured the resource in Deploy. See Configuring the resource in IBM DevOps Deploy on page 1677.
  - Created an application in Deploy. See Creating an application in IBM DevOps Deploy on page 1678.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Applications**.
   Result
   A list of applications that are available in Deploy is displayed.
3. Click the application that you want to configure from the **Name** column.
   Result
   The **Environments** page for the selected application is displayed.
4. Perform the following steps to create an environment for the application that you selected:

a. Click **Create Environment**.
b. Enter a name for the environment in the **Name** field.
c. Click **Save**.

5. Perform the following steps to configure resources to the environment:

   a. Click the environment that you created.

   b. Click **Add Base Resources**.
   Result
   A list of resources that are available in Deploy is displayed.

   c. Select the checkbox to add resources to the environment.

   d. Click **Save**.
   Result
   You can view the corresponding agent and the component that you added for the resource by using the **Expand** icon.

6. Perform the following steps to add the component to the application:
   a. Click **Applications**, and then select your application from the list.
   b. Click the **Components** tab, and then **Add Components**.
   c. Select the checkbox from the drop-down list to add components to the application.
   d. Click **Save**.
7. Perform the following steps to create a process for the application:

   a. Click the **Processes** tab, and then **Create Process**.

   b. Enter a name for the process in the **Name** field.

   c. Click **Save**.
   Result
   The **Design** tab for the process that you created is displayed.

8. Drag the component process listed under the **Component Process Steps** option from the left navigation pane and drop it into the design area.
9. Select the component process from the drop-down list in the **Operational (No Version Needed) Process** field.
10. Click **Save**.
11. Click the **Edit** icon, and then change the name of the properties.
12. Click **OK**, and then click **Save**.

Results

You have configured the application to run test assets from Deploy.

What to do next

You can run test assets from Deploy. See Running tests from IBM DevOps Deploy on page 1681.

Running tests from IBM® DevOps Deploy

You can run test assets from IBM® DevOps Deploy (Deploy) as a **Process** to test your application under test.

Before you begin

- You must be familiar with working with Deploy.
- You must have performed the following tasks:
  - Been granted access to Deploy.
  - Created a component in Deploy. See Creating a component in IBM DevOps Deploy on page 1670.
  - Created a process for the component in Deploy. See Creating a process in IBM DevOps Deploy on page 1671.
  - Configure the process for the component in Deploy. See Configuring the process in IBM DevOps Deploy on page 1672.
  - Created a resource in Deploy. See Creating a resource in IBM DevOps Deploy on page 1676.
  - Configured the resource in Deploy. See Configuring the resource in IBM DevOps Deploy on page 1677.
  - Created an application in Deploy. See Creating an application in IBM DevOps Deploy on page 1678.
  - Configured the application in Deploy. See Configuring the application in IBM DevOps Deploy on page 1679.
  - Started the Deploy agent. For more information, refer Starting agents.

1. Log in to Deploy, if you are not already logged in.
   Result
   The Deploy dashboard is displayed.
2. Click **Applications**.
   Result
   A list of applications that are available in Deploy is displayed.
3. Click the application that you configured for the test run.
4. Click the **Request Process** icon.
   Result
   The **Run Application Process** page is displayed.
5. Select the process that you created for the application which contains the test assets from the **Process** drop-down list.
6. Click **Submit**.
   Result
   The Deploy dashboard shows the progress of the process request.

Results

You have run the performance test assets as **Process** from Deploy.

What to do next

After the process completes, you can view the status of the process displayed as follows:

- **Success**: When the test run is successful
- **Failed**: When the test run is failed

You can view the details of the test run in the **Execution Log** section. The **Reports information** section on the Log file displays the names of the report along with its corresponding URLs.

Integration with IBM® Engineering Test Management

As a *Project Owner*, you can integrate Engineering Test Management as a quality management solution with Test . You can run test assets that are available in a project from Engineering Test Management.

The following topics provides more information about the integration of Test with Engineering Test Management.

Task flow for integration of IBM® Engineering Test Management

You must perform the following tasks in sequence to integrate Test with Engineering Test Management. For more information about the tasks, refer to the links provided in the table.

S/N	Tasks	More information
1	Create a project in Test.	Creating a project on page 1728
2	Install the required version of Engineering Test Management on your computer.	Engineering Lifecycle Management Documentation
3	Download the Engineering Test Management adapter, and then extract the contents of a compressed file. For more information about specific versions of plugin, see Integration plugin compatibility matrix on page 1639.	My HCLSoftware (MHS)
4	Add the adapter files that you downloaded to the installation directory of Engineering Test Management.	Adding the adapter files to IBM Engineering Test Management on page 1683
5	Configure the details of Engineering Test Management on Test.	Configuring IBM Engineering Test Management on DevOps Test on page 1684
6	Create a test script and a test case on Engineering Test Management to run the test assets.	Preparing test cases in IBM Engineering Test Management on page 1685

S/N	Tasks	More information
7	Run the test assets from Engineering Test Management.	Running tests from IBM Engineering Test Management on page 1687

Adding the adapter files to IBM® Engineering Test Management

You must add the adapter files to the installation directory to run test assets of Test from Engineering Test Management.

Before you begin

You must have completed the following tasks:

- Installed the latest version of Engineering Test Management from the Jazz portal on your computer. See System Requirements for DevOps Test 2026.06 (11.0.9).
- Downloaded the adapter (a compressed file) for Engineering Test Management from the My HCLSoftware (MHS).

  For more information about specific versions of plugin, see Integration plugin compatibility matrix on page 1639.

About this task

You must add the adapter files to the location where you installed Engineering Test Management to run test assets of Test.

1. Extract the compressed file that you downloaded.
2. Copy the `etm-adapter-updatesite` folder.
3. Go to the location where you installed Engineering Test Management on your computer.
4. Place the `etm-adapter-updatesite` folder in the `sites` folder of the installation directory of Engineering Test Management.
   For example, `D:\ELM\JazzTeamServer\server\conf\qm\sites`.
5. Copy the `dthub-profile.ini` file from the `provision_profiles` folder of the extracted package and place it in the `provision_profiles` folder of the Engineering Test Management installation directory.
   For example, `D:\ELM\JazzTeamServer\server\conf\qm\provision_profiles`.

Results

You have added the adapter files to Engineering Test Management.

What to do next

You must perform the following tasks:

- You must reset the Engineering Test Management server from the web console by using the **Request Server Reset** option.

  If you have administrator privileges, then you can access the **Request Server Reset** option from the following URL:

  ```
  https://<hostname>:<portnumber>/qm/admin?
  internal#action=com.ibm.team.repository.admin.serverReset
  ```

Note: You must replace the following values with the actual values in the URL:

- <hostname> with the hostname that you used when you installed Engineering Test Management.
- <portnumber> with the port number that is associated with an IP address of the hostname.

- Restart the Engineering Test Management server.
- Configure the details of Engineering Test Management on Test to establish a connection between the two products. See Configuring IBM Engineering Test Management on DevOps Test on page 1684.

Configuring IBM® Engineering Test Management on DevOps Test

You must configure Engineering Test Management on Test to establish a connection between the two products.

Before you begin

You must have completed the following tasks:

- Granted sufficient privileges to access **Project area** on Engineering Test Management.
- Created a project area in Engineering Test Management.
- Created a project in Test.

About this task

If you have multiple instances of Engineering Test Management, then you can configure all those instances on Test. If you have only one instance of Engineering Test Management, you can configure the same instance multiple times. However, you must choose a different **Project area** during the configuration of Engineering Test Management.

1. Log in to Test, if you are not already logged in.
2. Open a project from the team space.
3. Click **Manage > Integration**.
   The **Integration** page is displayed.
4. Click **Add IBM ETM Adapter** option within the **IBM Engineering Test Management (ETM)** section.
   The **Add IBM ETM Adapter** dialog is displayed.
5. Enter the following information in the **Add IBM ETM Adapter** dialog:

Fields	Actions
Server URL	Enter the URL of Engineering Test Management. For example, https://<hostname>:<portnumber>/qm
Username	Enter the username associated with Engineering Test Management.

Fields	Actions
Password	Enter the password associated with the username of Engineering Test Management.

6. Click **Validate**.
7. Select the project name that you created in Engineering Test Management from the **Project area** list.
8. Enter a name for the Engineering Test Management adapter in the **Adapter name** field.

   Note: You cannot provide an adapter name that already exists for the **Project area**.

9. Click **Save**.

Results

You have configured the details of Engineering Test Management on Test.

What to do next

You must create a test script and test case on Engineering Test Management to run the test assets of Test. See Preparing test cases in IBM Engineering Test Management on page 1685.

Preparing test cases in IBM® Engineering Test Management

You must create a test script and a test case on Engineering Test Management to run the test assets.

Before you begin

You must have completed the following tasks:

- Added the adapter files to the installation directory of Engineering Test Management. See Adding the adapter files to IBM Engineering Test Management on page 1683.
- Configured the details of Engineering Test Management on Test . See Configuring IBM Engineering Test Management on DevOps Test on page 1684.
- Generated an offline user token from Test. See Generating an offline token on page 1709.
- Copied the following values to be used during the preparation of a test script and a test case:
  - Branch name of the Git repository
  - Asset ID of the test that you want to run
  - **Optional**: Name of the environment if you want to run the API test suite.

1. Log in to Engineering Test Management, if you are not already logged in.
2. Open your project area from where you want to run the test assets.
3. Click **Construction**, and then click **Test Script** from the drop-down list.

4. Enter a name for the test script in the **Test Script** field.
5. Select **HCL DevOps Test Hub** from the **Type** drop-down list.
6. Enter the following test information:

Fields	Actions
Offline Token	Enter the offline token that you generated from Test.
Branch	Enter the name of the branch where your test assets are stored.
Asset ID	Enter an asset ID of the test asset that is available on the **Execution** page of Test. **Tip:** You can hover over the name of the test asset to view and copy the asset ID.
Environment	Enter a name of the API test environment. **Remember:** You must use this field only if you want to run the API test suites.

7. Click **Save**.
8. Perform the following steps to add a test script to a test case:

    a. Click **Construction > Test Case**.
    Result
    The **Test Cases** page is displayed.

    b. Enter a name for the test case in the **Test Case** field.

    c. Click **Test Scripts** from the left navigation pane.

    d. Click the **Plus** icon from the **Test Scripts** panel.
    Result
    The **Add Test Scripts** dialog is displayed.

    e. Select the checkbox to add the test script that you created to the test case, and then click **Add and Close**.

    f. Click **Save**.

Results

You have prepared Engineering Test Management.

What to do next

You can run the test assets of Test from Engineering Test Management. See Running tests from IBM Engineering Test Management on page 1687.

Running tests from IBM® Engineering Test Management

After you create a test case in Engineering Test Management, you can run the test. The test case includes the test scripts that contains test assets of Test .

Before you begin

You must have performed the following tasks:

- Prepared Engineering Test Management to run test assets of Test. See Preparing test cases in IBM Engineering Test Management on page 1685.
- Configured the details of Engineering Test Management on Test. See Configuring IBM Engineering Test Management on DevOps Test on page 1684.

1. Log in to Engineering Test Management.
2. Open the test case that you created in Engineering Test Management.
3. Click the **Play** icon , and then click **Run**.
   Result
   The **Run Test Case** dialog is displayed.
4. Click the **ChangeMachine** icon next to the **Machine** field to select the adapter that you configured on Test.
5. Select the required adapter name from the list, and then click **OK**.
6. Click **Finish** to start the test run.
   The Engineering Test Management dashboard displays the progress of the test case run.

   Note: If the dashboard does not display the progress of the test run correctly, then you must refresh the browser to view the correct status.

Results

You have run the test assets of Test from Engineering Test Management.

What to do next

You can perform the following tasks:

- Click **Show Result** to view information about the test results from Engineering Test Management when the test run is complete.
- Click **Link to Result** from **Results details** to view the test results from the **Results** page of Test. See Test results and reports on page 1501.

Deleting an IBM® Engineering Test Management adapter

You might want to delete an IBM® Engineering Test Management adapter when it is no longer needed.

Before you begin

You must have completed the following tasks:

- Been assigned the role of *Owner* of a project or *Owner* of the project in Test .
- Configured the details of Engineering Test Management on Test. See Configuring IBM Engineering Test Management on DevOps Test on page 1684.

1. Log in to Test and open the team space that contains your project.
2. Click **Manage > Integration**.
   The **Integration** page is displayed.
3. Go to the **IBM Engineering Test Management (ETM)** and within the **Configuration details of IBM ETM Adapters** section.

   You can view the Engineering Test Management adapters that are configured with the project on Test.
4. Identify the Engineering Test Management adapter that you want to delete, and then click the **Delete** icon .

   Note: If any of the test runs are in progress on Engineering Test Management, then those runs are aborted.
5. Click **Confirm** to delete the Engineering Test Management adapter from the project.

Results

You have deleted the Engineering Test Management adapter from the project on Test.

Integration with Jenkins

You can use the Test plugin to integrate Test with Jenkins. After you complete the integration, you can run tests assets that are available in a project of Test from a Jenkins server.

You must have installed Jenkins on your computer to install the Test Jenkins plugin. You can run the following command to start the Jenkins server to support UTF-8 character sets:

java -Dfile.encoding=UTF8 -jar jenkins.war

You can use either the Jenkins **Freestyle** project or the **Pipeline** project to run test assets from Jenkins. With **Freestyle** project, you can create build step from the Jenkins UI to run the test assets. Whereas the **Pipeline** project uses a simple text scripts based on the Groovy programming language. You can define pipeline scripts during the configuration of the **Pipeline** project by using any of the following methods:

- **Pipeline script**: In this method, you can directly enter a script in the field provided within the Jenkins UI or you can select a sample **Pipeline script** from the drop-down list.
- **Pipeline script from SCM**: In this method, you can create a script with a Groovy editor and then commit the script file into the Git repository.

For more information about the Jenkins Pipeline, refer to the Jenkins documentation.

Refer to the following topics to learn more about integrating Jenkins with Test.

Task flows for running test assets from Jenkins

You can perform certain tasks to run test assets from the Jenkins **Freestyle** project or the **Pipeline** project.

The following table lists the task flows for running test assets from the Jenkins **Freestyle** project:

Tasks	Go to...
Install the Test Jenkins plugin.	Installing the Test Jenkins plugin on page 1690
Import the CA certificate to your custom trust store.	Using a custom trust store on page 1691
Configure the **Freestyle** project.	Configuring the Freestyle project on page 1692
Run Test test assets on Jenkins.	Running tests from Jenkins on page 1702

You can perform the following tasks to run test assets from the Jenkins **Pipeline** project by using the sample script:

Tasks	Go to...
Install the Test Jenkins plugin.	Installing the Test Jenkins plugin on page 1690
Import the CA certificate to your custom trust store.	Using a custom trust store on page 1691
Create a pipeline script from the Jenkins UI.	Creating a pipeline script from Jenkins on page 1696
Configure the **Pipeline** project.	Configuring the Pipeline project by using the sample script on page 1700
Run Test test assets on Jenkins.	Running tests from Jenkins on page 1702

You can perform the following tasks to run test assets from the Jenkins **Pipeline** project by using the script from Source Code Management (SCM):

Tasks	Go to...
Install the Test Jenkins plugin.	Installing the Test Jenkins plugin on page 1690
Import the CA certificate to your custom trust store.	Using a custom trust store on page 1691

Tasks	Go to...
Create a pipeline script and commit the script to the Git repository.	For more information, refer to the Jenkins documentation.
Configure the **Pipeline** project.	Configuring the Pipeline project by using the script from SCM on page 1701
Run Test test assets on Jenkins.	Running tests from Jenkins on page 1702

Installing the Test Jenkins plugin

Before you can run the tests that are available in your Test project on the Jenkins server, you must install the Test Jenkins plugin on the Jenkins server.

Before you begin

You must have completed the following tasks:

- Installed Jenkins on your computer. For more information about installing Jenkins, refer to the Jenkins documentation.
- Downloaded the Test Jenkins plugin from the My HCLSoftware (MHS).

  For more information about specific versions of plugin, see Integration plugin compatibility matrix on page 1639.

1. Log in to the Jenkins server.
   Result
   The Jenkins dashboard is displayed.
2. Click **Manage Jenkins > Manage Plugins**.
3. Click the **Advanced** tab.
4. Click **Choose File** from the **Upload plugin** section to locate and **Open** the Test Jenkins plugin.
5. Click **Upload**.
   Result

   The Test Jenkins plugin is displayed in the **Installed** tab.

Results

You have installed the Test Jenkins plugin on the Jenkins server.

What to do next

You must import the CA certificate to your custom trust store to establish a trusted and secure connection between the Jenkins server and Test. See Using a custom trust store on page 1691.

Using a custom trust store

You must use a custom trust store in the Jenkins build to establish a trusted and secure connection between the Jenkins server and Test .

Before you begin

You must have completed the following tasks:

- Saved the Certificate Authority (CA) assigned to Test.

  Note: You can contact the administrator of Test if you do not have a copy of the CA.

- Installed Java on the computer from where you access the Test UI.

About this task

If the SSL certificate assigned to Test is signed by an internal CA, then you must download and import the CA to a custom trust store. You can then use the custom trust store in the Jenkins build step to establish a trusted and secure connection between the Jenkins server and Test.

Note: If the internal CA is already imported to the default trust store that is used by the Jenkins server, you need not use a custom trust store.

Restriction:

When you use Red Hat Enterprise Linux (RHEL) operating systems for Jenkins, you must run the Jenkins service with a user who has access to the custom trust store path to utilize the custom trust store feature. To change the Jenkins user, you must open the `/etc/sysconfig/jenkins` file and set the JENKINS_USER to the user who has access to the custom trust store path.

You can then run the following commands to set JENKINS_USER to another user who has access to the path of the custom trust store:

```
$JENKINS_USER= <username>
```

For example, `$JENKINS_USER= <user1>`

Note: You must ensure that the user account is available in the `/etc/passwd` file.

You can then run the following commands to change the ownership of the Jenkins folder:

```
chown -R username:username /var/lib/jenkins
```

```
chown -R username:username /var/cache/jenkins
```

```
chown -R username:username /var/log/jenkins
```

For example,

```
chown -R user1:user1 /var/lib/jenkins

chown -R user1:user1 /var/cache/jenkins

chown -R user1:user1 /var/log/jenkins
```

After the change of ownership is complete, you can run the following command to restart the Jenkins server:

```
/etc/init.d/jenkins restart
```

1. Locate the default trust store file (`cacerts` file) from the JRE directory.
   For example: `C:\Program Files\Java\jdk-18.0.1.1\lib\security`
2. Copy the `cacerts` file to a location of your choice on your computer.
3. Place the CA assigned to Test in the same location where you saved the `cacerts` file.
4. Run the following command from the command-line interface or terminal to import the CA of Test to custom trust store:

   ```
   keytool -import -trustcacerts -file {path to the downloaded CA with the file extension} -alias {label
    for the certificate} -keystore {path to the trust store cacerts file}
   ```

   For example,

   ```
   keytool -import -trustcacerts -file C:\Users\Desktop\cert\ca-master-hcl.crt -alias "otsca" -keystore C:
   \Users\Desktop\cert\cacerts
   ```

   Note: The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore {path to the trust store cacerts file}
   ```

   For example, `keytool -storepasswd -keystore C:\Users\Desktop\cert\cacerts`

Results

You have successfully imported the downloaded CA to the custom trust store.

What to do next

You can run test assets either from **Freestyle** or **Pipeline** project from the Jenkins server.

Configuring the Freestyle project

You must configure a **Freestyle** project to add a build step, and then run tests that are available in your Test project from Jenkins.

Before you begin

You must have completed the following tasks:

- Installed the Test Jenkins plugin on the Jenkins server. See Installing the Test Jenkins plugin on page 1690.
- Created a Jenkins **Freestyle** project.
- Generated an offline user token from Test. See Access management for DevOps Test on page 1709.

1. Open your Jenkins **Freestyle** project from the list, and then click **Configure**.
2. Click the **Build** tab, and then click **Add build step**.
3. Select the **HCL DevOps Test Hub test** option from the drop-down list.
4. Provide the details about the test run for the fields in the following table:

   The following table lists the required fields that you must provide to run the test from Jenkins:

Fields	Action
Name	Enter a name for the Jenkins build step.
Server URL	Enter the URL of Test. The format of the URL is as follows: *https://hostname*.
Offline Token	Enter the offline user token that you generated from Test.

   The following table lists the optional fields that you can provide to run the test from Jenkins:

Fields	Descriptions
Asset Type	Use this field to select the test asset type that you want to run from the **Asset Type** drop-down list.
Branch	Use this field to select the branch of the Git repository from the drop-down list. The **Branch** drop-down list displays the branches that are available in the corresponding project of Test. **Note:** After you select the branch from the **Branch** drop-down list, if you want to change the URL of Test, offline user token, or the project in the **Build** step, then you must close the build step. Then you must click **Add build step**, select **Run HCL DevOps Test Hub test**, and then enter the details.
Datasets	Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run.

Fields	Descriptions
	You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format: `/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv` **Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.
Labels	Use this field to add labels to test results when the test run is complete. You can add multiple labels to a test result separated by a comma. For example, *label1, label2* After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of Test.
Project	Use this field to select a project from the drop-down list. The **Project** drop-down list displays the projects that are available in the corresponding team space of Test. **Note:** The **Project** drop-down list displays the projects where you are an *Owner* or a *member* of Test. You must be an *Owner* or a *Tester* of the project that is available in the team spaces to run the tests from the Jenkins server.
Secrets	Use this field to select the Secret from the **Classification** drop-down list if you created secrets for your project to use Secrets at test run time. The **Secrets** field is mandatory only if you select one of the following options in the **Asset Type** field: ◦ **APISUITE** ◦ **APITEST** ◦ **APISTUB** ◦ **POSTMAN** The following message is displayed when you select any other asset type than preceding list: `You can select Secrets only if you are running tests of type APISUITE, APITEST, APISTUB, or POSTMAN.`

Fields	Descriptions
Team Space	Use this field to select the name of the team space from the drop-down list. The **Team Space** drop-down list displays the names of the team spaces if you are an *Owner* or a *member* of the team space.
Test	Use this field to select the required test that you want to run from the drop-down list. The **Tests** drop-down list displays the available test assets from the corresponding branch in the following format: `Project_name [Path: the path of the test assets] [Repo: URL of the Git repository that the test belongs to]` **Note:** After you select the test details from the **Tests** drop-down list if you want to change the URL of Test, offline user token, or the project in the Jenkins **Build step**, then you must delete the build step and add a new build step.
Test Environment	Use this field to select an environment from the drop-down list. The **Test Environment** field displays the available test environments from Test. The **Test Environment** field is mandatory only if you select one of the following options in the **Asset Type** field: ◦ **APISUITE** ◦ **APITEST** ◦ **APISTUB** ◦ **POSTMAN** The following message is displayed when you select any other asset type than preceding list: `You can select Test Environment only if you are running tests of type APISUITE, APITEST, APISTUB, or POSTMAN.`
Use Custom Trust Store	Select this checkbox if you used an internal CA certificate and imported the certificate to the custom trust store. You can then enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field. Select the **Use Custom Password for the Trust Store** checkbox if you modified the trust store password, and then enter the new password in the **Custom Trust Store Password** field.

Fields	Descriptions
	Note: If you did not modify the trust store password, you must clear the **Use Custom Password for the Trust Store** checkbox.
Variables	Use this field to enter the name of the variable and its value if your test requires variables during the test run time. You must enter the variables in the following format: `name_of_the_variable=value_of_the_varibale` You can add multiple variables to the test run separated by a semicolon. For example, *varname1=value1;varname2=value2*
Fail the job if the test or any of the containing tests fail	Select the checkbox to mark the job as failed when the test or any child test within the test fails.

Note: You can run multiple tests sequentially in the same job by adding multiple build steps and providing details for the tests that you want to run.

5. Click **Save** to save the build step.

Results

You have configured the **Freestyle** project by adding the build step.

What to do next

You can run test assets from the Jenkins server. See Running tests from Jenkins on page 1702.

Creating a pipeline script from Jenkins

You must create a pipeline script to run test assets that are available in the Test project from the Jenkins **Pipeline** project.

Before you begin

You must have completed the following tasks:

- Installed the Test Jenkins plugin on the Jenkins server. See Installing the Test Jenkins plugin on page 1690.
- Generated an offline user token from Test. See Access management for DevOps Test on page 1709.
- Created a Jenkins **Pipeline** project.

1. Open your Jenkins **Pipeline** project from the list.
2. Click **Configure**, and then select the **Pipeline** tab.
3. Click **Pipeline Syntax** to generate a pipeline script.
4. Select the **step: General Build Step** option from the drop-down list in the **Sample Step** field.
5. Select the **Run HCL DevOps Test Hub test** option from the drop-down list in the **Build Step** field.
6. Provide the details about the test run for the fields in the following table:

   The following table lists the required fields that you must provide to run the test from Jenkins:

Fields	Action
Name	Enter a name for the Jenkins build step.
Server URL	Enter the URL of Test. The format of the URL is as follows: *https://hostname*.
Offline Token	Enter the offline user token that you generated from Test.

Field	Description
Asset Type	Use this field to select the test asset type that you want to run from the **Asset Type** drop-down list.
Branch	Select the branch from the drop-down list. The **Branch** drop-down list displays the branches that are available in the corresponding project of Test.
Datasets	Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run. You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format: `/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv`

Field	Description
	Note: You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.
Labels	Use this field to add labels to test results when the test run is complete. You can add multiple labels to a test result separated by a comma. For example, *label1, label2* After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of Test.
Project	Use this field to select a project from the drop-down list. The **Project** drop-down list displays the projects that are available in the corresponding team space of Test. **Note:** The **Project** drop-down list displays the projects where you are an *Owner* or a *member* of Test. You must be an *Owner* or a *Tester* of the project that is available in the team spaces to run the tests from the Jenkins server.
Secrets	Use this field to select the Secret from the **Classification** drop-down list if you created secrets for your project to use Secrets at test run time. The **Secrets** field is mandatory only if you select one of the following options in the **Asset Type** field: ◦ **APISUITE** ◦ **APITEST** ◦ **APISTUB** ◦ **POSTMAN** The following message is displayed when you select any other asset type than preceding list: `You can select Secrets only if you are running tests of type APISUITE, APITEST, APISTUB, or POSTMAN.`
Team Space	Use this field to select the name of the team space from the drop-down list. The **Team Space** drop-down list displays the names of the team spaces if you are an *Owner* or a *member* of the team space.

Field	Description
Test	Use this field to select the required test that you want to run from the drop-down list. The **Tests** drop-down list displays the available test assets from the corresponding branch in the following format: `Project_name [Path: the path of the test assets] [Repo: URL of the Git repository that the test belongs to]` **Note:** After you select the test details from the **Tests** drop-down list if you want to change the URL of Test, offline user token, or the project in the Jenkins **Build step**, then you must delete the build step and add a new build step.
Test Environment	Use this field to select an environment from the drop-down list. The **Test Environment** field displays the available test environments from Test. The **Test Environment** field is mandatory only if you select one of the following options in the **Asset Type** field: ◦ **APISUITE** ◦ **APITEST** ◦ **APISTUB** ◦ **POSTMAN** The following message is displayed when you select any other asset type than preceding list: `You can select Test Environment only if you are running tests of type APISUITE, APITEST, APISTUB, or POSTMAN.`
Use Custom Trust Store	Select this checkbox if you used an internal CA certificate and imported the certificate to the custom trust store. You can then enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field. Select the **Use Custom Password for the Trust Store** checkbox if you modified the trust store password, and then enter the new password in the **Custom Trust Store Password** field. **Note:** If you did not modify the trust store password, you must clear the **Use Custom Password for the Trust Store** checkbox.
Variables	Use this field to enter the name of the variable and its value if your test requires variables during the test run time.

Field	Description
	You must enter the variables in the following format: `name_of_the_variable=value_of_the_varibale` You can add multiple variables to the test run separated by a semicolon. For example, *varname1=value1;varname2=value2*
Fail the job if the test or any of the containing tests fail	Select the checkbox to mark the job as failed when the test or any child test within the test fails.

7. Click **Generate Pipeline Script** and copy the script that you created for the build step to use it in the sample script.

Results

You have created the pipeline script.

What to do next

You can configure the **Pipeline** project by using the **Pipeline script** option. See Configuring the Pipeline project by using the sample script on page 1700.

Configuring the Pipeline project by using the sample script

You can configure a **Pipeline** project directly from the Jenkins UI by using the sample pipeline scripts to run test assets from Jenkins.

Before you begin

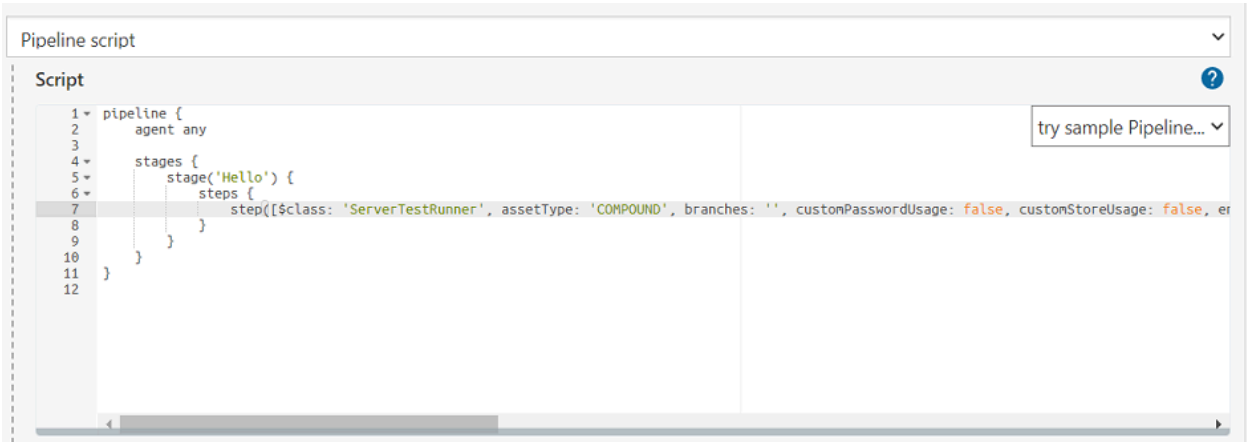
You must have completed the following tasks:

- Installed the Test Jenkins plugin on the Jenkins server. See Installing the Test Jenkins plugin on page 1690.
- Created a pipeline script. See Creating a pipeline script from Jenkins on page 1696.

1. Open your Jenkins **Pipeline** project from the list.
2. Click **Configure**, and then select the **Pipeline** tab.
3. Select the **Pipeline script** option in the **Definition** field.
4. Select the sample script from the drop-down list.

   For example, if you select `Hello Word` from the drop-down list, then the sample script is displayed as follows:

5. Specify the script that you created in place of *"echo 'Hello Word'"*.



Note: You can provide a different name for the `stage` in the script.

Results

You have configured the **Pipeline** project by using the **Pipeline script** option.

What to do next

You can run test assets from the Jenkins server. See Running tests from Jenkins on page 1702.

Configuring the Pipeline project by using the script from SCM

You can configure the **Pipeline** project to use the pipeline script from the Source Code Management (SCM), and then run test assets that are available in the Test project from the Jenkins **Pipeline** project.

Before you begin

You must have completed the following tasks:

- Installed the Test Jenkins plugin on the Jenkins server. See Installing the Test Jenkins plugin on page 1690.
- Created a pipeline script and committed the script to the Git repository. For more information, refer to the Jenkins documentation.
- Been granted administrator permission of the Jenkins server to add credentials into Jenkins.
- Added your GitHub credentials into Jenkins. For more information about adding global credentials, refer to the Jenkins documentation.
- Created a Jenkins **Pipeline** project.

About this task

If your pipeline scripts are complex, you can then write the script outside of Jenkins UI. You must commit that script file into your Git repository. During the **Pipeline** build process, Jenkins checks out the script file from the Git repository and runs your **Pipeline** project.

1. Open your Jenkins **Pipeline** project from the list.
2. Click **Configure**, and then select the **Pipeline** tab.
3. Select the **Pipeline script from SCM** option in the **Definition** field.
4. Select the **Git** option from the drop-down list in the **SCM** field.
5. Enter the URL of the Git repository in the **Repository URL** field where you stored your pipeline script.
6. Select the credentials of the Git repository from the drop-down list.
7. **Optional:** Enter the branch name of the Git repository in the **Branch Specifier** field.
8. Enter the path of the script that you stored in the Git repository in the **Script path** field.
9. Click **Save**.

Results

You have configured the **Pipeline** project by using the **Pipeline script from the SCM** option.

What to do next

You can run test assets from the Jenkins server. See Running tests from Jenkins on page 1702.

Running tests from Jenkins

You can run test assets either from the Jenkins **Freestyle** or the **Pipeline** project on the Jenkins server to test an application under test.

Before you begin

You must have completed the following tasks:

- Configured the **Freestyle** project, if you want to run test assets from the **Freestyle** project. See Configuring the Freestyle project on page 1692.
- Configured the **Pipeline** project either by using the **Pipeline script** or **Pipeline script from SCM**, if you want to run test assets from the **Pipeline script** project. See Configuring the Pipeline project by using the sample script on page 1700 or Configuring the Pipeline project by using the script from SCM on page 1701.
- Downloaded the latest **Test Results Analyzer** plugin from the Jenkinsci repository. For more information about the **Test Results Analyzer** plugin, refer to the Jenkins documentation.
- Installed the **Test Results Analyzer** plugin on the Jenkins server. For more information about installing the plugin, refer to the Jenkins documentation.

About this task

After the test run is complete, you can view the test results in a tabular format when you use the Jenkins **Test Results Analyzer** plugin. With a TRA-based report, you can filter the results based on the status of the build. The status of the builds can be PASSED, FAILED, and SKIPPED. You can also identify the failed build from the TRA-based report and take the required action to resolve the build errors. The TRA report also represents the build information in **Line**, **Bar**, and **Pie** graphs so that you can easily read the data.

1. Log in to the Jenkins server.
   Result
   The Jenkins dashboard is displayed.
2. Open your Jenkins **Freestyle** or **Pipeline** project from the list.
3. Click **Build Now** to run the test assets from Jenkins.

Results

You have run the test from the Jenkins server.

What to do next

You can view the build logs by clicking the build number from the **Build History** pane, and then selecting the **Console Output** option. You can also view the history of test results in a tabular format by clicking the **Test Results Analyzer** option.

You can access the report URLs to view the test run information at any point in time. You can also view the test reports and logs of the test that was run on the Jenkins server from the **Results** page on Test. See Test results and reports on page 1501.

Integration with Xray test management for Jira

When you use Jira as a change management system on Test and you use Xray as the test management application in Jira, you can integrate Xray with Test. With this integration, you can export the test results of the tests that run on Test to your Jira project. You can then view and use the test results in your project in Jira.

Prerequisites

Before you integrate Xray with Test, you must have completed the following tasks:

- Configured Jira as a change management system on Test. See Configuration of Atlassian Jira as a change management on page 1609.
- Installed the Xray test management app in Jira.

  Note: The Xray integration is supported only when the Xray test management application is installed on a Jira cloud.

- Ensured that you obtained the access and authentication details to Jira from the Jira administrator.
- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

Task flow

You can follow the tasks that are listed in the following table when you want to export test results to your Xray test project in Jira. The table also provides you the links to the information about the tasks.

Task	Go to...
Generate the Xray API keys for your username on Jira.	Generating API keys for the Xray app in Jira on page 1704
Use the generated Xray API keys to configure the Xray integration in Test.	Configuring the Xray integration on page 1705
Select an issue in your Xray project in Jira as a location to export the test results of a test run. You must select the issue while you configure a test run on Test.	Selecting a Jira Xray issue to export test results on page 1706

You can manage the Xray settings that you configured in Test. You can edit the saved configuration or you can delete the configuration. For more information, see Managing the Xray configuration on page 1708.

Generating API keys for the Xray app in Jira

You must generate API keys for the Xray app in Jira and use them to configure the Xray integration in Test . The Xray API keys are used by Jira to authenticate the requests from Test.

Before you begin

You must have completed the following tasks:

- Read about the prerequisite tasks in Prerequisites on page 1704.
- Verified that you can access Jira.

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.

1. Log in to Jira.
2. Click **Settings > Apps**.

   The apps that are installed in Jira are listed in the navigation pane.
3. Find the **Xray App** in the navigation pane, and then click the **API Keys** option.

   The **API Keys** page is displayed.
4. Click **Create API Key**.

   The **Create API Key** dialog is displayed.
5. Enter your username in the **User** field, and then select the username that is displayed.
6. Click **Generate**.

   The API keys are generated and displayed under the **Client ID** and **Client Secret** fields.
7. Click the **Copy** icon in the **Client ID** field to copy the key, and then paste the copied key into a text file.
8. Click the **Copy** icon in the **Client Secret** field to copy the key, and then paste the copied key into a text file.

Results

You have generated the Xray API keys in Jira.

What to do next

You must use the Xray API keys to configure the Xray integration in Test. See Configuring the Xray integration on page 1705.

Configuring the Xray integration

You can set up the Xray integration when you want to export test results from Test to an issue in your Xray project in Jira. You must configure the Xray integration in your project in Test before you select the Xray project in Jira as the location to export the test results after the test runs in Test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that you must have set up Jira as the Change Management system to add Jira Xray Integration.
- Generated the Xray API keys in Jira and saved the keys in a text file. See Generating API keys for the Xray app in Jira on page 1704.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.
4. Click **Add Jira Xray Integration** option within **Jira Xray Test Management** section.

   The **Add Xray Integration** dialog is displayed.
5. Perform the following steps:

   a. Enter the Xray API key that was generated for the **Client ID** in the **Client ID** field.

   b. Enter the Xray API key that was generated for the **Client Secret** in the **Client Secret** field.

   c. Click **Save**.

      The Xray configuration is saved and displayed under the **Jira Xray Test Management** pane.

Results

You configured the Xray integration on Test.

What to do next

You can perform the following tasks:

- View the following information on the **Jira Xray Test Management** that you used to configure the Xray integration:
  - The URL of the Jira server on which the Xray test management app is installed.
  - The project that you selected in Jira.
  - The Xray API key that is used as the Client ID.
- Select an issue in Jira as the location to which you can export the test results of a test run while you configure a run of the test on Test. See Selecting a Jira Xray issue to export test results on page 1706.

Selecting a Jira Xray issue to export test results

After you configure the Xray integration on Test , you can select a Jira Xray issue in Jira or create a new issue in Jira to export the test results. The test results are exported to the selected issue after the tests run on Test. You can then view and track the test results in your Jira project.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Configured the Xray integration by using the Xray API keys that are generated for your username. See Configuring the Xray integration on page 1705.
- Ensured that the **Execute test asset** dialog is displayed and you have configured the other settings for a test run in Test.

About this task

When you configure a test run in Test, you can select a Jira issue as the location where you want the test results to be exported. After the test is run, the results are exported to the Jira issue. You can then view and track the test results in the Jira issue from your project in Jira.

1. Click the **RESULTS** tab in the **Execute test asset** dialog.

   The **RESULTS** tab is displayed.

   Note: The **Export report options** field is enabled only if you configured the Xray integration with the Xray API keys that are generated for your username in Jira.

2. Perform the actions listed in the following table to select an option to export the test results:

Export options	Results in...	Actions
Jira Xray	The test results of the test that you configure for a run are exported to the selected issue after the test run is completed. The results are exported to the selected issue that exists in your Xray project in Jira.	Perform the following steps: a. Select the **Jira Xray** option. The **Select issue** field is displayed. b. Click the field to view the list of issues in the Xray project in Jira. Alternatively, you can enter the issue key in the Jira Xray project. c. Select the issue to which you want to export the test results. d. Continue with the next step.
New Jira item	The test results of the test that you configure for a run are exported to a new Jira issue. The issue is created in your project in Jira after the test run is completed.	Perform the following steps: a. Select the **New Jira item** option. b. Continue with the next step.

Export options	Results in...	Actions
None	The test results are not exported. **Note:** This export option is the default option.	Perform the following steps when: ◦ You do not want to export the test results. ◦ You selected a Jira issue in a previous run of the same test, and you do not want to export the results of the current run. a. Select the **None** option. b. Continue with the next step.

Results

You selected an issue in your Xray project or created an issue in Jira as a location to export the test results of a test run on Test.

What to do next

After the test run is completed, you can go to the **Results** page and expand the test result to view the Xray issue key to which the test results were exported. You can then click the Xray issue key link to open the issue and view the details of the issue in Jira.

Managing the Xray configuration

You can manage the configuration of the Xray integration that you set in Test when you want to modify the API keys configured or when you want to delete the existing Xray configuration.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to the server projects on page 1735.
- Ensured that you obtained the access and authentication details to Jira from the Jira administrator.
- Configured the Xray API keys for the Xray integration on Test. See Configuring the Xray integration on page 1705.
- Regenerated the Xray API keys in Jira by following the steps to generate the Xray API keys in Generating API keys for the Xray app in Jira on page 1704.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Integration** in the navigation pane.
   The **Integration** page is displayed.

4. Go to the **Jira Xray Test Management** and within the **Configuration details of Jira Xray** section the details about the Xray integration is configured are displayed.
5. Perform any of the following actions when you want to update or delete the Xray configuration:
    - To update the Xray API keys that were set previously in the Xray configuration, go to Step 6 on page 1709.
    - To delete the Xray configuration, go to Step 7 on page 1709.
6. Perform the following steps to update the Xray API keys:

    a. Click the **Update Credentials** icon of the Xray integration that you want to manage.
    The **Update Xray Integration** dialog is displayed.

    b. Enter the Xray API key that was generated for the **Client ID** in the **Client ID** field.

    c. Enter the Xray API key that was generated for the **Client Secret** in the **Client Secret** field.

    d. Click **Update** to save the updated Xray API keys in the Configuration details of Jira Xray.

    The Xray configuration is updated and a confirmation message is displayed.
7. Perform the following steps to delete the existing Xray configuration:

    a. Click the **Delete** icon.
    The **Delete Xray configuration** dialog is displayed.

    b. Click **Confirm**.

    The Xray configuration is deleted and a confirmation message is displayed.

Note: The links to the issues that you created to export the test results are not deleted from the results panel in Test even after you delete the Xray configuration.

Results

You achieved the following results:

- Updated the Xray configuration.
- Deleted the Xray configuration.

Access management for DevOps Test

Desktop clients and third-party integrations use offline user tokens to connect to Test .

You can configure desktop clients and third-party integrations to access Test by using the offline user tokens generated from the server.

Note: Offline tokens remain valid indefinitely. However, if a token is unused for 30 days, it expires. When you use an offline token, it uses the permissions that are currently assigned to your user account.

You can use offline user tokens to perform the following tasks:

- Retrieve secrets configured on the server to be used in certain tests in HCL DevOps Test Integrations and APIs (Test Integrations and APIs).
- Enable publishing of test results and reports from desktop products that include Test Integrations and APIs, HCL DevOps Test Performance (Test Performance), or HCL DevOps Test UI (Test UI) to Test.
- Enable the Resource Monitoring Service in Test Performance to monitor a data source during a run on Test.
- Enable agents to connect with Test.
- Enable HTTP proxies to connect with Test.
- Enable third-party integrations.

See the following table for a list of integrations for which you can use the tokens.

Integrations	More information
Test Integrations and APIs with Test	See Test Integrations and APIs and Test.
Test Performance with Test	See Publishing test results to the server. See Enablement of Resource Monitoring services for a schedule.
Test UI with Test	See Publishing test results to the server.
Third-party integrations	See Integrations with other applications on page 1639.

Creating an offline user token in DevOps Test

You can create an offline user token, copy it, and then use that token to enable connections to be authenticated by Test .

Before you begin

You must have logged in to the Test with an active user account.

1. Click the **User** icon from the menu bar, and then select **Create Token**.

   The **Create Offline User Token** dialog is displayed with a token.

   Note: After you copy the token, it is no longer accessible because the dialog closes.

2. Click **Copy** to copy the token.

   The dialog closes.

3. Paste the copied token in a text file and save it for reference.
4. Use the token in the UI of the desktop products or in the application where this token is required.

Results

You have created an offline user token.

Deleting an offline user token in DevOps Test

You can delete all the offline tokens when you want to revoke server access or when the existing tokens are no longer required.

Before you begin

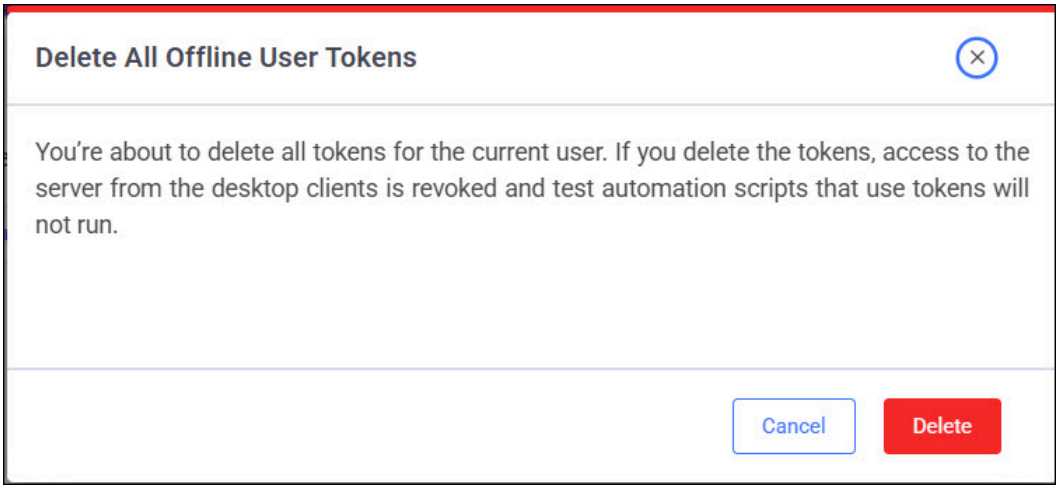
You must have logged in to Test with an active user account.

About this task

Offline tokens provide access for desktop clients and third-party integrations to connect to the server.

Warning: Deleting offline tokens invalidates all existing offline tokens. Any desktop client, automated agent, proxy, or third-party integration that currently uses these tokens will lose connection to the server and fail to authenticate until a new token is configured.

1. Click the **User** icon from the menu bar, and then select **Delete Tokens**.
   The **Delete All Offline User Tokens** dialog is displayed with a warning.



2. Click **Delete** to confirm deletion.
3. Perform any of the following necessary actions:
   - Inform the affected desktop client users that the existing tokens were deleted.
   - Update any test automation scripts that used the deleted tokens to prevent authentication failures.

What to do next

If you need to restore connections for desktop applications, agents, or integrations, you must complete the task to generate a new offline user token. See Creating an offline user token in DevOps Test on page 1710.

Working with DevOps Test REST services

When you want to use the REST APIs of Test in your test automation scripts, you can view the REST APIs that are exposed by Test. You can also run the APIs from within your project in Test.

All REST services are accessible from the base URL of Test and are defined by using Swagger. A navigable user interface lists the available set of APIs at `<Test_Hub_URL>/test/#/Initial/Admin/docs`. This Swagger file describes the individual APIs along with details of the parameters and response bodies.

You must be assigned the role of an administrator to be able to view or run the REST APIs.

After you log in to Test with your credentials that has administrator privileges, you must go to the following URL: `<Test_Hub_URL>/test/#/Initial/Admin/docs`

The Swagger docs that are available can be viewed from the drop down list.

You can select the Swagger documentation for the REST APIs that you want to view or use. For example, you can select the **Execution API (Public)** to view the APIs for the execution micro-service.

Note: Only the Execution API is public, and other APIs are internal and subject to change. The Swagger UI indicates whether an API is considered as public.

You can perform the following steps to view or run the API:

1. Expand the section to view the APIs that are available.
2. Expand each of the APIs to view the details such as the description, examples, parameters, and responses.
3. Click **Try it out**.
4. Enter the values for the required parameters.
5. Click **Execute** to run the API.

You can view the response after the API runs.

You can view the API doc for the execution micro-service that is extracted in this PDF document.

Related information

Running APIs in command line on page 1712

Running APIs in command line

You can run the APIs in the command line by adding a bearer access token to authorization header in every API call.

About this task

To generate the bearer access token, you must create an offline token.

1. Click the **User** icon , and select **Create Token** to obtain an offline token.

   Result

   The token is automatically generated and displayed in the **Create Offline User Token** dialog.
2. Click **Copy** to copy the token and save it in a secure location.

   You may assign the token to variable `OFFLINE_TOKEN` for using in the following steps.
3. Generate an access token by sending a POST request to `<baseUrl>/test/rest/tokens/` with the post data below:

   `refresh_token=$OFFLINE_TOKEN`

   Result

   The response JSON displays an 'access_token' field.
4. Add HTTP header Authorization with the access token in every API call:

   Authorization: `Bearer $access_token`

   The following example uses a bash command to start the execution:

```
TEST_HUB_URL=UNSET_URL
PROJECT_ID=UNSET_PROJECT_ID
BRANCH=UNSET_BRANCH
ASSET_PATH=UNSET_ASSET_PATH
if [ -n "$OFFLINE_TOKEN" ] && [[ "$PROJECT_ID" != *"UNSET"* ]] && [[ "$TEST_HUB_URL" != *"UNSET"* ]] &&
 [[ "$ASSET_PATH" != *"UNSET"* ]]; then \
  CURL_CMD="curl -k"; \
  echo "Getting access token from $TEST_HUB_URL/test/rest/tokens/"; \
  access_token=$($CURL_CMD -H "Accept: application/json" \
  -d refresh_token=$OFFLINE_TOKEN $TEST_HUB_URL/test/rest/tokens/ | grep -oP
 '"access_token"\s*:\s*"\K[^"]*'); \
  POST='{"testAsset": {"revision": "'$BRANCH'", "name": "'$ASSET_PATH'"},"offlineToken":
 "'$OFFLINE_TOKEN'"}'; \
  echo "Posting execution to $TEST_HUB_URL/test/rest/projects/$PROJECT_ID/executions/"; \
  execution_rsc=$($CURL_CMD -H "Content-Type: application/json" -H "Authorization: Bearer $access_token"
 \
        -d "$POST" $TEST_HUB_URL/test/rest/projects/$PROJECT_ID/executions/); \
  echo $execution_rsc; \
else \
  echo "OFFLINE_TOKEN env variable is not set (UI->Account->Create Token) or PROJECT_ID/ASSET_PATH is
 not set"; \
fi
```

Results

You ran APIs outside of the Swagger UI by using an offline token.

Test Manager Guide

This guide describes how to manage and track your overall test effort. This guide is intended for test managers.

Related information

Managing access to the server projects on page 1735

Managing team spaces

You might want to become a team space member, add members to your team space, change a user role in a team space, modify the team space, or delete a team space.

Each team space is associated with at least one *Team Space Owner*. You can assign multiple owners to your team space. Any user who becomes a member of a team space and can be assigned the role of a *Member*, *Project Creator*, *Architect*, or *Team Space Owner* of a team space, can access the team space.

As an owner of a team space, after you create a team space, you must configure the team space license. You can then add members to the team space and can assign different roles to each member of the team space. You can update or delete a team space. You can also add a repository to a team space to store a system model and edit the repository settings. Apart from adding a repository, you can add, update, or delete the Resource Monitoring agent if no longer required.

As an *Architect* of a team space, you can manage a team space repository and design a system model.

As a *Project Creator* of a team space, you can create a project in the team space.

You can view the configured agents or intercepts of your team space.

Viewing team spaces

After you log in to Test , you can view all the existing team spaces on the **Team Space Dashboard** page.

About this task

As a member of a team space, you can view your team spaces under **My Team Spaces**. You can also view the team spaces that are created by other users as the public team space and the Permissive team spaces under **Other Team Spaces**.

If the initial team space does not exist, then you can view the **Team Space Dashboard** without any navigation pane.

You as a member of a team space, when you want to use that team space immediately after you log in to Test, you can suffix the alias in the application URL.

For example, if the alias of the team space is `first`, then you must use the following Test URL in the address bar:

```
<server url>/#/first
```

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane, and then click **Join Team Space**.
   Result
   The **Team Space Dashboard** is displayed.

   You can view the team spaces in the following panels:

- **My Team Spaces**: You can view all the team spaces that you created. You can also view the team spaces in which you are a member. You can create a new team space by clicking **New Team Space** option.
- **Other Team Spaces**: You can view all the team spaces that are created by others.

Note: You cannot view the Restrictive private type team spaces.

Results

You have viewed the team spaces that are created by you and other users on the **Team Space Dashboard**.

What to do next

You can select a team space and can send a request to become a member of that team space. See Becoming a team space member on page 1720.

Adding members to a team space

When you create a team space to manage projects on Test , you must add users to your team space so that they can create and access the projects.

Before you begin

- As an administrator, you are assigned the default role of a *Team Space Owner*. If you log in to Test as another user, then you must have been assigned the role of a *Team Space Owner*.
- As an owner of a team space, you must have completed the following tasks:
  - Created a team space. See Creating a team space on page 745.
  - Configured a license for the team space. See Configuring licenses for team spaces.

You must have been assigned a role as a *Team Space Owner*. If you are a server administrator, then you are assigned with the default role of a *Team Space Owner*.

About this task

As an owner of a team space, you can add multiple users to your **Permissive** or **Restrictive** team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane.

   Note: If the initial team space does not exist, then you can view the **Team Space Dashboard** page without any navigation pane.

3. Identify the team space for which you want to add a member.
   You can either use the same team space where you choose to work or switch to any other team space.
4. Click **Manage > Configuration**.

5. Click **Add member** to add members in the initial team space.

   Result

   The **Add Member** page is displayed.
6. Enter a username in the **Add Members** dialog to add users.

   Result

   The list of users is displayed.
7. Select the user.
8. Assign a role to the selected user.

Note: All members are assigned the default roles of a *Project Creator* and *Architect*.

You can choose the role for the user based on its associated actions:

Roles	Responsibilities
Team Space Owner	◦ Create a team space. ◦ Update and delete a team space. ◦ Add members to a team space. ◦ Remove members from the team space. ◦ Assign the roles to the members. ◦ Modify the roles of the members. ◦ Create and modify a project. ◦ Add or update the system model. ◦ Add a repository to store a system model. ◦ Update and delete a repository in a team space. ◦ Configure or update agent and intercepts. ◦ Register, update, delete, and list Resource Monitoring sources in a team space.
Project Creator	◦ Create, update, and view a project ◦ Configure or update agent and intercepts.
Architect	◦ Create, update, delete, and view a repository in a team space. ◦ Create, update, and delete components in the system model. ◦ Configure or update agent and intercepts.
Member	◦ View the list of team spaces in the **Team Space Dashboard.** ◦ View the configuration of the team space. ◦ Configure or update agent and intercepts. ◦ View the registered agents, or intercepts. ◦ View the Resource Monitoring agents in a team space. ◦ View the registered Resource Monitoring agents. ◦ View components in the system model.

Notes:

- The members with the assigned role of a *Team Space Owner*, *Project Creator*, *Architect*, or *Member* of a team space can be assigned with any of the following project roles:
  - *Project Owner*
  - *Tester*
  - *Viewer*
- You can add or remove the role of a member by using the **Edit roles for members** icon next to each user in the **Members** list.

9. Repeat the applicable steps to add another user.

Results

You have added members with assigned roles to your team space. You can view the list of all added users on the **Team Space Configuration** page.

What to do next

You can reassign or remove the roles of users to one or more team spaces that you own. See Managing members and their roles in a team space on page 1721.

All users can request to be a member of another team space. See Becoming a team space member on page 1720.

Users added as members to the team space can view the team space in **My Team Spaces**.

Modifying the configuration of a team space

After you create a team space on Test , if you want to modify the details of the team space, add a repository to your team space, add or remove members of a team space or delete a team space, then you can perform these tasks from the **Team Space Configuration** page.

Before you begin

You must have been assigned the role of a *Team Space Owner*.

About this task

The **Team Space Configuration** page displays details of the team space.

As an owner of a team space, you can modify details of the team space, add members to your team space, and assign roles to the members of the team space on the **Team Space Configuration** page.

The *Team Space Owner* and *Architect* can add or remove a repository from the team space.

If the team space is no longer required, then the *Team Space Owner* can also delete that team space.

As a *Member*, *Project Creator*, or *Architect*, you can only view the configuration details of a team space on the **Details** tab of the **Team Space Configuration** page.

As a *Member*, and *Project Creator* you can only view the configuration details of a team space on the **Repository** tab of the **Team Space Configuration** page.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane.
3. Identify the team space for which you want to modify the configuration.
   You can either use the same team space where you are working or switch to any other team space.
4. Click **Manage > Configuration**.
   Result
   The **Configuration** page is displayed.
5. Perform the modifications on the following details of the **Configuration** page of the team space:

Fields	More information
Name	Displays the name of the team space.
URL alias	Displays an alias of the team space. You must suffix this alias in the application URL to access a specific team space after you log in to the application.
Member Access	Displays two options for Member Access:: ◦ Approval required to join the team space. ◦ Team space can be discovered by potential members.:
Description	Brief description of the team space.
Members	List of members in a team space.
Repositories	Displays the details of the repository that host the system model.

Results

You have modified the configuration details of the selected team space.

What to do next

You can add members and assign roles to each member of your team space.

Related information

Adding members to a team space on page 1715

Managing members and their roles in a team space on page 1721

Adding repositories to a server project on page 1729

Viewing the configuration of a team space

You can view the configuration of a team space as a member of any team space on Test . You can view the configuration details on the **Team Space Configuration** page.

Before you begin

You must be a member of a team space.

About this task

The **Team Space Configuration** page displays the configuration details of the team space.

As a member of a team space, if you want to access your team space immediately after logging into Test, then you must suffix the alias in the application URL. You can find details of the alias on the **Team Space Configuration** page. You can also view the list of members and their roles in that team space on the **Team Space Configuration** page.

1. Log in to Test.
2. Click the **Switch Team Space** icon from the left navigation pane.
3. Identify the team space for which you want to view the configuration.
   You can either use the same team space where you are working or switch to any other team space.
4. Click **Configuration** under **Manage** tab.
   Result
   The **Configuration** page is displayed.

   You can view the following details about the team space:

Fields	Tabs	More information
Name	**Details**	Displays the name of the team space.
URL alias	**Details**	Displays an alias of the team space. You must suffix this alias in the application URL to access a specific team space after you log in to the application.
Member Access	**Details**	Displays two options for Member Access::

Fields	Tabs	More information
		◦ Approval required to join the team space. ◦ Team space can be discovered by potential members.:
Description	**Details**	Displays a brief description of the team space.
Members	**Details**	Displays a list of members of the team space.
Repositories	**Details**	Displays the details of the repository that host the system model.

Results

You have viewed the configuration details of the selected team space.

Becoming a team space member

After logging into Test , the licensed users must be a member of any team space to perform test activities in a project. As a licensed user, if you are not assigned the role of a member in any team space, then you can request the *Team Space Owner*to become a member of any existing team space.

Before you begin

You must be a licensed user.

About this task

If you want to become a member of a **Restrictive** team space, then only the owner of that team space can grant you the role of a member of a team space. If you request to become a member of a **Permissive** team space, then you automatically become a member of that team space.

Note: The server administrator can join any team space automatically without any approval and is assigned with a default role of *Team Space Owner*.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane, and then **Join Team Space**.
   Result
   The **Team Space Dashboard** is displayed.

   You can view the team spaces in the following panels:

- **My Team Spaces**: You can view all the team spaces that you created. You can also view the team spaces in which you are a member.
- **Other Team Spaces**: You can view all the team spaces that are created by others.

Note:

You can view only the public type of **Restrictive** team spaces and the **Permissive** team spaces. You cannot view the **Restrictive** private type team spaces.

3. Select the team space that you want to join from the list of other team spaces in the **Team Space Dashboard**.
4. Click the **Key** icon of the selected team space.
5. Confirm your request when prompted.

If you agree, then the *Team Space Owner* sees a notification about the request.

After opening the notification, the *Team Space Owner* can view the user who is requesting the access and can accept or decline the request. If the *Team Space Owner* accepts your request, you are added and assigned a *Member* role.

Notes:

- You can become a member of any team space only when the license of the selected team space is configured.
- If you are a server administrator, then the *Team Space Owner* role is the default role assigned.
- If you are a licensed user, then the *Member* role is the default role assigned and is restricted to specific actions.

You can then see the team space under **My Team Spaces** on the **Team Space Dashboard** page.

You might want to follow up with the *Team Space Owner* if your request is pending for some time.

Results

You have become a member of another team space.

What to do next

You can view the actions that you can perform based on your assigned role. See Adding members to a team space on page 1715.

Managing members and their roles in a team space

After you, as a *Team Space Owner* assign roles to the members of a team space onTest , you might want to remove a member from the team space or reassign the role to the existing members of your team space.

Before you begin

You must have been assigned the role of a *Team Space Owner*.

About this task

As a licensed user, you are assigned a role of a *Member* in any team space after the approval of the *Team Space Owner*.

As an owner of a team space, you can remove or reassign any member of your team space with the roles of a *Team Space Owner*, *Project Creator*, or *Architect*. You can also remove any member from the list of members of the team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane.
3. Select the team space for which you want to reassign the roles of the members or remove a member from the list of members.
   You can either use the same team space where you are working or switch to any other team space.
4. Identify the member from the list of **Members**.
5. Perform any one of the following operations to the identified member of the team space:
   Choose from:
   - Click **Remove member** to remove the identified member from the team space.
   - Click **Edit roles for member** and select the role that you want to assign or remove the assigned role for the identified member.

     You can choose multiple roles for the members based on the associated actions:

Roles	Actions
Team Space Owner	▪ Create a team space. ▪ Update and delete a team space. ▪ Add members to a team space. ▪ Remove members from the team space. ▪ Assign the roles to the members. ▪ Modify the roles of the members. ▪ Create and modify a project. ▪ Add or update the system model. ▪ Add a repository to store a system model. ▪ Update and delete a repository in a team space. ▪ Configure or update agent and intercepts. ▪ Register, update, delete, and list Resource Monitoring sources in a team space.
Project Creator	▪ Create, update, and view a project. ▪ Configure or update agent and intercepts.

Roles	Actions
Architect	▪ Create, update, delete, and view a repository in a team space. ▪ Create, update, and delete components in the system model. ▪ Configure or update agent and intercepts.
Member	▪ View the list of team spaces in the **Team Space Dashboard.** ▪ View the configuration of the team space. ▪ Configure or update agent and intercepts. ▪ View the registered agents, or intercepts. ▪ View the Resource Monitoring agents in a team space. ▪ View the registered Resource Monitoring agents. ▪ View components in the system model.

Note: The *Team Space Owner*, *Project Creator*, *Architect*, or *Member* of a team space can be assigned with any of the following project roles:

- *Project Owner*
- *Tester*
- *Viewer*

Similarly, you can remove the assigned role of any member of your team space.

Results

You have completed any of the following tasks in your team space:

- Reassigned the role of the members.
- Removed the assigned role of the members.
- Removed the members.

Related information

Adding members to a team space on page 1715

Deleting a team space

You might want to delete a team space on when it is no longer needed.

Before you begin

You must have been assigned the role of a *Team Space Owner* of Test .

About this task

As a *Team Space Owner*, you can delete a team space that you do not need. When you delete a team space, it is a permanent deletion. You cannot undo the delete action on a team space.

If you delete the initial team space, then you can only view the **Team Space Dashboard** page without any navigation pane.

After you delete the team space, you can use the name of the team space again.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Switch Team Space** icon from the left navigation pane.
3. Identify the team space that you want to delete.
   You can either use the same team space where you are working or switch to any other team space.
4. Click **Manage > Configuration**.
   Result
   The **Configuration** page is displayed.
5. Click the **Delete Team Space** icon .
6. Confirm that you want to delete the team space when prompted.

   If you proceed, the team space is deleted including all of its projects and members.

Results
You have deleted the selected team space.

What to do next
You can view the available team spaces on the **Team Space Dashboard** page. See Viewing team spaces on page 1714.

After you delete a team space, the license that is used for that team space is released and is available to configure the other team spaces, if required. See Configuring licenses for team spaces.

Managing DevOps Test projects

Test projects manage access to your test assets, which are stored in a Git repository. Projects are maintained in a team space. Projects are either public by default or private. Private projects are not discoverable by other users. You can either add your own project or you can request to be a member of another public project.

All the licensed users are assigned with a default role of a *Project Creator*. As a *Project Creator*, when you create a project immediately after logging in the application, the project is created in the initial team space. If you want to create a project in any specific team space, then you must become a member of that team space. When you add your own project, you can configure it now or later. Configuring the project includes adding details about the project, adding one or more Git repositories, optionally adding secrets, classifying encrypted datasets, and adding a change management system. To

run test assets that are associated with an encrypted dataset, you must categorize the encrypted datasets by creating a classification. You can also add users to your project to access your test assets only if the users are the members of that team space. To use Jira to create defects for the tests available in your project, you must configure the Jira server for your project on Test.

Related information

Team space overview on page 742

Becoming a team space member on page 1720

Becoming a project member on page 1735

Managing access to the server projects on page 1735

Managing repositories on page 1741

Protecting API test assets by using secrets on page 1749

Managing an encrypted dataset on page 976

Default user administration

Configuring Jira as a change management on page 1610

Working with projects in a team space

After you log in to Test , you must have a project to perform the test activities. To work with projects, you must create and configure projects.

Before you begin

You must have logged into Test.

About this task

You must be assigned with the role of an *Owner*, *Tester*, or *Viewer* of a project.

1. Go to the **Projects** page from the **Initial Team Space** Home page. See Viewing active projects on page 1733.
2. Add a project. See Creating a project on page 1728.
3. Add a repository. See Adding repositories to a server project on page 1729.
4. Add secrets in a classification. See Secrets configuration on page 1732.
5. Encrypt the data set resources. See Managing an encrypted dataset on page 976.
6. Configure a change management system. See Configuration of a change management on page 1608.
7. Add members to a project and configure their roles. See Adding users to a project on page 1732.
8. Select a global branch in a project to view the test assets or resources maintained in the repositories. See Selecting the global branch in a project on page 1748
9. Perform the following tasks based on the type of test assets or resources that you want to run:

a. Add agents to your project, if you are using remote agents as a location to run certain tests. See Adding a performance agent to a project on page 1076.
b. Read the considerations that you must take into account before you configure a run or start a virtual service. See Prerequisites for running HTTP virtual services on page 1455.
c. Read the considerations that you must take into account before you configure a test run. See Prerequisites to running tests on page 1057.

10. Configure a run of the test asset or resource. See Test run configurations on page 1134.
11. View the progress of a test run. See Viewing the progress of running test assets on page 1444.
12. Manage a running test. See Management of running tests on page 1440.
13. View the results of a test run after the run is completed. See Test results on page 1499.
14. Review the test results and create defects for test runs from the Results page. See Creating Jira issues for the test results on page 1612.

Results

You have performed all the tasks with the projects.

Repository considerations for a server project

To collaborate with other project stakeholders, you can open test assets from a local clone of a remote repository, pull project test assets from the remote repository, and push changes made to your local test assets to the remote repository. Before you add a project and then add a repository to that project, you must consider some information about repositories.

Consider the following sections about version control systems that are used to manage remote repositories.

The following types of Git-based version control systems are supported:

- GitHub, which is an open-source product for storing and collaborating code.
- Azure Repos, which is part of a solution of Azure DevOps, and focuses on collaboration between DevOps professionals to develop private code.

Both these version control systems support Git repositories to store code.

You must be familiar with the various aspects of these version control systems to manage the application code that you want to test in Test .

You can learn more about installing, setting up, and using Git from the Git documentation.

You can learn more about installing, setting up, and using Azure Repos from the Azure Repos documentation.

Note: Git is the only tool that is supported to manage a repository in Azure Repos. Other tools, which can be used in Azure Repos, are not supported.

Git

You must install Git or upgrade to the supported version if you already have installed Git. See System Requirements for DevOps Test 2026.06 (11.0.9).

Repositories and user identities

After you install Git, you must set up your Git repository and set up access for members. You must ensure that the repository contains your test assets.

Optionally, you can use a command line utility or Git tool to access the repository, upload your test assets, fetch or pull from the repository, push to the repository, clone the repository, and other operations that you want to perform in Git.

Local and shared repositories

After you create a remote or shared repository in Git, you can create a local version of the repository by cloning the remote repository. You must ensure that your test assets are available in the remote repository and are also cloned to the local repository.

Alternatively, if your test assets are on your local system, you can set up a Git repository in the bare mode, add the project files to the local repository, and then commit and push from the local repository to the remote repository in Git by using your preferred method.

Note: While copying the test assets from your local system to the repository, you must ensure that you copy the entire project that contains the test assets.

User authentication for the Git repository

The administrator can set up different types of authentications for accessing the Git repository. Test supports the following authentication types:

- HTTP with username and password
- HTTP without username and password
- HTTPS with username and password
- HTTPS without username and password
- SSH with SSH key and passphrase
- SSH with SSH key and without a passphrase

Based on the authentication type that is set for a repository, you must provide the same authentication values in Test when you add a repository.

Test assets

You must complete the following tasks in the desktop products where you author your test before you check in and commit the test assets to the Git repository.

Test type	Task	More information
API suite in Test Integrations and APIs	Change the local stub to a remote stub.	See Test run considerations for API Suites or API tests on page 1059.
	Add the library files.	

Creating a project

You can create a project and enter its details.

Before you begin

To create and manage projects for your test assets, you must log in to Test by providing the application URL in a browser.

If you are a new user, and LDAP and Active Directory are not configured, then you must first sign up by filling out a form with your email, username, and password. You can then log in by using those details.

You must be a member of any licensed team space.

About this task

As a licensed user, you are by default assigned as both *Project Creator* and *Architect* role for the initial team space. You must select a team space to create a project.

You can view all the team spaces in the **Team Space Dashboard**. To create a project in any team space, you must be a member of that team space and assigned a *Team Space Owner* or *Project Creator* role in that team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **New Project** button on the **Projects** page.
   Result
   The **New Project** dialog is displayed.
3. Assign a name to your project.
4. **Optional:** Enter the description of the project.
5. Click **Create**.

Results

You have created a project.

What to do next

You can modify your project configuration. You can also add a repository to your project.

Related information

Default user administration

Changing the visibility configuration of a project on page 1734

Adding repositories to a server project on page 1729

Adding repositories to a server project

You can add a single repository individually or multiple repositories simultaneously to a server project to access assets and resources that are available in the repositories.

Before you begin

You must have completed the following tasks:

- Added a project on Test .
- Been granted permission and you must have set up your credentials to access the remote repositories in a repository domain.
- Created test assets and resources that are contained in either a single git repository or multiple git repositories.

About this task

As a project member with the role of a *Project Owner*, you can add a single repository or multiple repositories to your project.

Note: You can only add a single repository that is managed by using Git in Azure Repos. The addition of multiple repositories in Azure Repos is not supported.

You can use the information in the following table to decide the method for adding repositories to your project:

When...	Action...
You want to add any of the following repositories: • A single repository owned by you. • A single repository owned by other people. • The repositories that are in different domains.	You can add individual repositories one at a time. You must then provide the complete URL of the repository along with the required credentials to authenticate the connection to the repository from Test. For example, the URL to a single repository can be as `https://github.com/<user>/<repository>.git`. You must provide the *username* and *password* to access this repository.
You want to add any of the following repositories:	You can add all the repositories simultaneously that are in the same domain.

When...	Action...
• Multiple repositories that are owned by you and are contained in the same domain. • Multiple repositories that are owned by others and are contained in the same domain.	You must then provide the URL of the repository domain along with the required credentials to authenticate the connection to the repository from Test. For example, the URL of the repository domain can be `https://github.com`. You must select the platform and provide your *personal access token* to access the repositories.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the navigation pane or from the **My Projects** page.

   The **Overview** page of the project is displayed.
3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.
4. In the **Repositories** section, click **Add repository**.

   The **Add repository** dialog is displayed.

   You can either choose to add a single repository individually or multiple repositories simultaneously from a single domain.
5. Perform either of the following actions:
   - To add a single repository, go to Step 6 on page 1730.
   - To add multiple repositories from a domain, go to Step 7 on page 1731.
6. Perform the following steps to add individual repositories at a time:

   a. Enter the complete URL of the repository in the **Git URL** field.

      For example, the URL to a single repository can be as `https://github.com/<user>/<repository>.git`.

      The required authentication fields are displayed that depend on the protocol you specified in the URL.

If...	Then...
The repository URL contains *http* or *https*	The following fields for authentication are displayed: ▪ **Username** ▪ **Password**
The repository URL contains *git@*	The following fields for authentication are displayed:

If...	Then...
	▪ **SSH/Deploy Key** ▪ **Passphrase** You can add a Git repository by using git@ and SSH. For example: ▪ `git@github.com:account/testRepo.git` ▪ `ssh://git@10.111.111.333:2222/~/account/project.git`

Note: If you use SSH to connect to a remote repository and Test displays an `Auth Fail` error, you can resolve this error by regenerating your SSH keys by using the **-m PEM** option.

b. Enter the credentials to authenticate the connection to the repository.

c. Click **Add** to add the repository to your project.

You can add other individual repositories by specifying the URL and performing the actions in Step 6 on page 1730.

7. Perform the following steps to add multiple repositories simultaneously that are contained in the same domain:

Note: You can only add a single repository that is managed by using Git in Azure Repos. The addition of multiple repositories in Azure Repos is not supported.

a. Enter the URL of the repository domain in the **Git URL** field.

For example, the URL of the repository domain can be `https://github.com`.

b. Select the platform from the list under **Platform** where your repositories reside.

c. Enter your *personal access token* to authenticate your connection to the repository domain.

Note: If you do not have your *personal access token*, you can click **Generate** to generate your *personal access token* and use the token to authenticate your connection.

d. Click **Next**.

The connection to the repository domain is established and all the repositories that you own are listed. The time when the repository was last updated is displayed in the **Last updated** column.

e. Select the repositories that you want to add to your project.

f. Click **Add**.

Note: Depending on the size of the repository that you are adding, it can take a few to several minutes to add the repository to your project on the server.

All the repositories that you selected are added to your project and are listed on the **Repositories** page.

Note: If the repository is read-only, you will be able to add it and view its assets, but you will not be able to publish changes directly to it from Test Hub.

Results

You have added individual repositories or multiple repositories to your project on the server.

What to do next

You can perform the following actions on the repository that you added:

- Update the authentication credentials if they are changed in the Git repository configuration. See Updating the authentication credentials of the repository on page 1745.
- Delete a repository if it is no longer required. See Deleting a repository on page 1745.
- Refresh a repository to fetch and synchronize changes from the remote repository. See Refreshing repositories manually on page 1746.

Secrets configuration

If you are working with an API suite and the project test asset contains environment variables that are required for the test runs, you must configure the environment variables as secrets in a classification by using the **Add Secret** option in the **Security** page. Configuring secrets enable the API suite to run correctly on Test.

Related information

Protecting API test assets by using secrets on page 1749

Adding users to a project

You can add users to your project in Test so that they can access your test assets.

Before you begin

- You own a project.
- The users you want to add to your project must be the members of the team space of that project.

About this task

As a project owner, you can add other users to your public or private project. For example, if you have a large number of test assets that must be run, you might want your test team to help run them, and see the results.

1. Log in to Test.
   Result
   The **Projects** page of the Initial Team Space is displayed.
2. Select the project for which you want to add the users.
3. Click the **Configure Project** icon on the Project card.

   The **Configuration** page of the project is displayed.
4. In the **Project Members** section, click **Add Members**.

   The **Add Members** dialog is displayed.
5. Enter the name or email of the user in the search bar to find the user.
6. Select **Viewer**, **Tester**, or **Owner** role for the user that you want to add.

   The added user after logging in can see the owner's project in their list of **My Projects**.
7. Click **Add Members**.

   The newly added members are listed under the **Project Members** list.

Results

You have successfully added users to your project.

What to do next

You can add or remove roles to one or more projects that you own. All users can request to be a member of another public project.

Related information

Becoming a project member on page 1735

Managing access to the server projects on page 1735

Viewing active projects

After you log in to Test , you can view both active and archived projects in the **Initial team space**.

About this task

As a project owner, you can view your projects under the **My Projects** tab, and the projects that are created by other users as public projects are displayed under the **Other Projects** tab. You can view a list of all active projects that you created under **My Projects**. You can view a list of all archived projects by selecting the **Include Archived Projects** check box.

Log in to Test.

The **Projects** page is displayed.

You can view the following projects:

- The active projects that you created or added to as a member with either the *Tester* or *Viewer* role are displayed under the **My Projects** tab.
- The active projects that are created by others as public projects are displayed under the **Other Projects** tab.

Results

You have viewed the active projects that are created by you and other users in the team space.

What to do next

You can perform any of the following tasks:

- Create a project. See Adding details to a server project on page 1728.
- Request to become a member of a project that is displayed under the **Other Projects** tab. See Becoming a project member on page 1735.
- View the other pages in the team space and perform tasks on those pages. See Tasks in a team space on page 747.

Related information

Archiving or unarchiving server projects on page 1738

Changing the visibility configuration of a project

After you create a project in Test , you might want to change the visibility configuration of the project from the public, which is the default, to private.

Before you begin

You must have created a project.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Select the project from **My Projects** tab.
3. Click **Manage > Configuration**.
   Alternatively, select a project, and then click the **Configure project** icon on the project card.
   Result
   The **Configuration** page is displayed.
4. In the **Visibility** section, select **Private** as the visibility mode of the project.
5. Click **Save**.

Results

You have successfully changed the visibility configuration of a project.

What to do next

You can add a repository to your project.

Related information

Default user administration

Becoming a project member

You can request to be a member of another project. New users without any projects can also request to be a member of an existing project.

About this task

As a project owner, you can add users to your public or private projects. All users can request to become a member of a public project.

Note: A project owner can add a licensed user as a member of the project only if that user is the member of the team space.

1. Log in to Test.
   Result
   The **Projects** page of the initial team space is displayed.
2. Click the **Other Projects** tab and search for the project that you want to join.
3. Click the Project card or click **Join Project**.
   Only public projects are visible in the list of projects.
4. Confirm your request when prompted. If you agree, the project owner is notified.
5. The project owner sees a pending request **Notification** icon on their Project card.

   After opening the notification, the owner can see the user that is requesting access and can accept or decline it.

   If the project owner accepts your request, you are added as a member of the project with a **Viewer** role. A **Viewer** role is the default role assigned and is restricted to specific actions.

   You then see the project under **My Projects** tab.

   If the project owner declines your request, follow up with the project owner.

Results

You have sent a request to become a member of a project.

Managing access to the server projects

You might want to add or remove a project member role from your project.

About this task

As a project owner, you can assign access by specifying a role when you add a user to your project. You can also assign more access or remove access for a user. Roles enable users to perform tasks on project resources such as running tests and viewing test results.

1. Choose the role in a project based on its associated actions.

Option	Description
Owner of a project	◦ All Tester's actions ◦ Update, archive, list, and view a project ◦ Assign, update, delete, and list user roles for a project ◦ Accept new member requests ◦ Create, update, and delete a dataset classification in a project ◦ Add, update, and remove an encrypted dataset from a classification ◦ Update current row for a listed dataset in a project ◦ Delete, list and view a report ◦ Update, delete, and list a repository in a project ◦ Get the test assets from a repository in a project ◦ Create, update, delete and list secrets in a project ◦ Create, update, delete, and list resource monitoring sources in a project ◦ Get the content of a resource monitoring source in a project ◦ Register, update, delete, and list resource monitoring agents ◦ Create, configure, and delete the Jira change management system ◦ Create and update defects in the Jira change management system ◦ Create and update components in the system model
Tester of a project	◦ All Viewer's actions ◦ List repositories in a project ◦ Run a test and create a report ◦ Stop a test while running ◦ Get the test assets from a repository in a project ◦ Get a list of datasets in a project ◦ Get the content of a dataset

Option	Description
	◦ Update the current row for a listed dataset in a project ◦ Add, update, and remove an encrypted dataset from a classification ◦ Delete a report if you created the report ◦ Create and update defects in the Jira change management system ◦ Create and update components in the system model
Viewer of a project	◦ List your own roles on a project ◦ List running tests ◦ List and view a report ◦ List resource monitoring sources in a project ◦ Get the content of a resource monitoring source in a project ◦ List the resource monitoring agents ◦ View components in the system model ◦ View defects in the Jira change management system
Non-member of a project	◦ Request to be a member of a project

2. Open the project that you own.
   From the **Configuration** page, you can see a list of members in your project along with their roles and then you can modify their roles and add or remove the members from your project.
3. Click **Manage > Configuration** from the navigation pane.
   From the **Configuration** page, you can see a list of members in your project along with their roles and then you can edit their roles and add or remove the members from your project.
4. To edit the roles of the existing user, perform the following steps in the **Project Members** section:
   a. Hover over the user for whom you want to modify the role and then click the **Edit roles for members** icon.
   b. Select the new role for the user and then click the **Done editing members role** icon to save the modified role for the user.
5. To delete the roles of the existing user, perform the following steps in the **Project Members** section:
   a. Hover over the user for whom you want to modify the role and then click the **Remove member from the project** icon.
      The **Remove Member** dialog is displayed.
   b. Click **Yes** to remove the user from the project.

Archiving or unarchiving server projects

You can archive a project when that project is no longer needed but you want to retain it for future reference. If you do want to use an archived project, you can unarchive it.

About this task

As a project owner, you can archive inactive projects in Test . Archived projects are not visible to users or project members, but owners can show or hide their archived projects. You can view the list of all archived projects from the navigation pane. Archived projects can be unarchived.

- To archive a project:
  1. Log in to Test.
  2. Select your project from the **Projects** page.
  3. Click the **Configure project** icon on the project card.
     Result
     The **Configuration** page is displayed.
  4. Click the **Archive Project** icon.
     The **Archive Project** dialog is displayed.
  5. Click **Archive** to confirm that you want to archive the project.
     If you proceed, the project is archived and is hidden from **My Projects**.
- To view an archived project:
  1. Log in to Test.
     Result
     The **Projects** page of the initial team space is displayed.
  2. Select **Include Archived Projects** check box.
     Result
     A list of all the projects including archived projects that are associated with you is displayed under **My Projects** and the archived projects that are associated with others are displayed under **Other Projects**.
- To unarchive a project:
  1. Log in to Test.
     Result
     The **Projects** page of the initial team space is displayed.
  2. To view archived projects, check the box labeled **Include Archived Projects**.

3. Select your archived project, and then click the **Configure project** icon on the project card.
   Result
   The **Configuration** page is displayed.

4. Click the **Unarchive Project** icon.

   The **Unarchive Project** dialog is displayed.

5. Click **Unarchive** to confirm that you want to unarchive the project.
   If you proceed, the project is unarchived and is shown under **My Projects** tab.

Deleting server projects

You might want to delete a project when the project is no longer needed, and you want to free up disk space.

About this task

As a project owner, you can delete a project that you no longer want. When you delete a project it is permanent. You cannot undo the delete project action.

After you delete the project, you can use the project name again.

1. Open the project that you want to delete.
2. From the **Configuration** page, click the **Delete Project** icon.

   The **Delete Project** dialog is displayed.
3. Select the **I understand project deletion is permanent.** checkbox.

   Warning:

   If you proceed, the project is deleted including all of its test assets, results, repository and members.

4. Click **Delete** to confirm that you want to delete the project.

Results

You have deleted a project.

An overview of test assets, modifications, and scheduled runs

After you configure a project with one or more Git repositories and run some test assets, you can view statistics about those test assets.

From the **Overview** page, several cards display charts, sliders, and tables of the statistics of the test assets in your server project. You can click some of the charts, sliders, or tables to see more information, such as the progress of a scheduled

run or the results of a run. You can also review some statistics by date and see more details such as the proportion of test assets by type.

You can view a server project, which might contain multiple repositories and multiple branches, by selecting the name of the branch from the **Branch** drop-down list. For more information about selecting branches, see Selecting the global branch in a project on page 1748.

The **Members** area displays the total number of users who have access to the project. You can click the number to view the details of all the members. The **Owner** area displays the username of the project owner.

Test suites

This card displays a slider of the percentage of test suites by type. A test suite is a collection of tests, including schedules, Compound Tests, API Suites, and AFT Suites. You can hover over the slider and see the proportion of test suites by test suite type. The number of suites shown are relative to the selected branch.

Individual tests

This card displays a slider of the percentage of individual tests by type. You might have performances tests, API tests, or UI tests according to the desktop clients that you used to create the tests. You can hover over the slider and see the proportion of individual tests by type. The number of tests shown are relative to the selected branch.

Execution results

This card displays a chart of the proportion of test suites that were run by status and by date. If no date is selected, the card displays all the run results associated with the project since it was created. You can see the run results by clicking a test or Suite status. You can also hover over the chart and see the test or Suite status and the proportion of test assets. You can view statistics by tests or Suites that ran.

Notes:

- The results shown are generated from tests or Suites run.
- The statistics exclude canceled runs of tests or Suites.
- The number of Suites and individual tests that are associated with your project are shown on the **Home** page along with a verdict summary. You can see the run results by clicking a verdict. You can also see the last date and time when you ran one or more test assets.
- The number of suites shown are relative to the selected branch.

Last run

This card displays a table of the last three runs by date and time. You can see the run results by clicking each last run.

New or modified

This card displays a chart of new and modified test assets that were made in your Git repository for a week or for the last seven days based on a selected date range. You can hover your mouse over each bar in the chart and see the proportion of test assets by type. The data shown is relative to the selected branch.

Scheduled runs

This card displays a table of the next three scheduled runs by date and time. You can see the progress of a scheduled run by clicking each scheduled run.

Insights

This card displays a table of the insights on the supported parameters identified from the tests that were run in the project. The table lists the level of confidence with which the analysis of the specific parameter was done, the parameter analyzed, the name of the test asset, and the timestamp of the completed analysis. You can click any item in the row of a specific insight to view the details of that insight on the **Insights** page.

Related information

Managing DevOps Test projects on page 1724

Managing repositories

After you create a project in Test , you can add repositories that contain test assets and resources to the project. You can also change the repository details, update the user credentials of a repository, delete a repository, create a webhook for the configured repository.

Tasks in a repository

To add a repository in your project and work on the repository, you can perform the tasks listed in the following table:

Task	More information
Adding a repository	Adding repositories to a server project on page 1729
Updating the repository credentials	Updating the authentication credentials of the repository on page 1745
Refreshing a repository	Refreshing repositories manually on page 1746
Creating a webhook for the repository	Creating webhooks on page 1749
Selecting the global branch	Selecting the global branch in a project on page 1748
Deleting a repository	Deleting a repository on page 1745

Adding repositories to a server project

You can add a single repository individually or multiple repositories simultaneously to a server project to access assets and resources that are available in the repositories.

Before you begin

You must have completed the following tasks:

- Added a project on Test .
- Been granted permission and you must have set up your credentials to access the remote repositories in a repository domain.
- Created test assets and resources that are contained in either a single git repository or multiple git repositories.

About this task

As a project member with the role of a *Project Owner*, you can add a single repository or multiple repositories to your project.

Note: You can only add a single repository that is managed by using Git in Azure Repos. The addition of multiple repositories in Azure Repos is not supported.

You can use the information in the following table to decide the method for adding repositories to your project:

When...	**Action...**
You want to add any of the following repositories: • A single repository owned by you. • A single repository owned by other people. • The repositories that are in different domains.	You can add individual repositories one at a time. You must then provide the complete URL of the repository along with the required credentials to authenticate the connection to the repository from Test. For example, the URL to a single repository can be as `https://github.com/<user>/<repository>.git`. You must provide the *username* and *password* to access this repository.
You want to add any of the following repositories: • Multiple repositories that are owned by you and are contained in the same domain. • Multiple repositories that are owned by others and are contained in the same domain.	You can add all the repositories simultaneously that are in the same domain. You must then provide the URL of the repository domain along with the required credentials to authenticate the connection to the repository from Test. For example, the URL of the repository domain can be `https://github.com`. You must select the platform and provide your *personal access token* to access the repositories.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the navigation pane or from the **My Projects** page.

The **Overview** page of the project is displayed.

3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.

4. In the **Repositories** section, click **Add repository**.

   The **Add repository** dialog is displayed.

   You can either choose to add a single repository individually or multiple repositories simultaneously from a single domain.

5. Perform either of the following actions:

   - To add a single repository, go to Step 6 on page 1743.
   - To add multiple repositories from a domain, go to Step 7 on page 1744.

6. Perform the following steps to add individual repositories at a time:

   a. Enter the complete URL of the repository in the **Git URL** field.

      For example, the URL to a single repository can be as `https://github.com/<user>/<repository>.git`.

      The required authentication fields are displayed that depend on the protocol you specified in the URL.

If...	Then...
The repository URL contains *http* or *https*	The following fields for authentication are displayed: ▪ **Username** ▪ **Password**
The repository URL contains *git@*	The following fields for authentication are displayed: ▪ **SSH/Deploy Key** ▪ **Passphrase** You can add a Git repository by using git@ and SSH. For example: ▪ `git@github.com:account/testRepo.git` ▪ `ssh://git@10.111.111.333:2222/~/account/project.git`

Note: If you use SSH to connect to a remote repository and Test displays an `Auth Fail` error, you can resolve this error by regenerating your SSH keys by using the **-m PEM** option.

b. Enter the credentials to authenticate the connection to the repository.

c. Click **Add** to add the repository to your project.

You can add other individual repositories by specifying the URL and performing the actions in Step 6 on page 1743.

7. Perform the following steps to add multiple repositories simultaneously that are contained in the same domain:

Note: You can only add a single repository that is managed by using Git in Azure Repos. The addition of multiple repositories in Azure Repos is not supported.

a. Enter the URL of the repository domain in the **Git URL** field.

For example, the URL of the repository domain can be `https://github.com`.

b. Select the platform from the list under **Platform** where your repositories reside.

c. Enter your *personal access token* to authenticate your connection to the repository domain.

Note: If you do not have your *personal access token*, you can click **Generate** to generate your *personal access token* and use the token to authenticate your connection.

d. Click **Next**.

The connection to the repository domain is established and all the repositories that you own are listed. The time when the repository was last updated is displayed in the **Last updated** column.

e. Select the repositories that you want to add to your project.

f. Click **Add**.

Note: Depending on the size of the repository that you are adding, it can take a few to several minutes to add the repository to your project on the server.

All the repositories that you selected are added to your project and are listed on the **Repositories** page.

Note: If the repository is read-only, you will be able to add it and view its assets, but you will not be able to publish changes directly to it from Test Hub.

Results

You have added individual repositories or multiple repositories to your project on the server.

What to do next

You can perform the following actions on the repository that you added:

- Update the authentication credentials if they are changed in the Git repository configuration. See Updating the authentication credentials of the repository on page 1745.
- Delete a repository if it is no longer required. See Deleting a repository on page 1745.
- Refresh a repository to fetch and synchronize changes from the remote repository. See Refreshing repositories manually on page 1746.

Updating the authentication credentials of the repository

If the authentication credentials of the Git repository which was added to your project are changed, then you must update the credentials of the repository in Test .

Before you begin

You must have completed the following tasks:

- Been assigned a *Member* or *Project Creator* role in a team space.
- Been assigned the *Owner* role in the project with access permissions to edit repositories.

About this task

In Test, the repository is refreshed at regular intervals to pull the latest changes committed to the Git repository. After you update your authentication credentials of the repository, the repository is refreshed to ensure that you have access to the Git repository and to the latest assets in the repository.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the left navigation pane or from the **My Projects** page.
3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.

4. In the **Repositories** section, click the **Change...** icon under Actions column for the repository that you want to update.

   The **Edit Repository** dialog is displayed.

5. Enter the new credentials, and then click **Update**.

   The new credentials are updated and your access to the repository is retained.

Deleting a repository

You can delete repositories that you no longer require in your test environment.

Before you begin

You must have completed the following tasks:

- Configured the repository that contains the test assets in your project on Test .
- Been assigned a *Owner* role with access to delete repositories.

About this task

You can delete a repository that you have configured for your project.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the left navigation pane or from the **My Projects** page.
3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.
4. In the **Repositories** section, click the **Delete** icon under **Actions** column for the repository that you want to update.

   The **Delete repository** dialog is displayed.
5. Select the checkbox, and then click the **Delete** button.

   The repository is deleted and removed from the list.

Refreshing repositories manually

You can manually trigger the refresh activity for a repository to instantly fetch and synchronize the changes from the remote repository.

Before you begin

You must have completed the following tasks:

- Configured the repository that contains the test assets in your project onTest .
- Been assigned a *Member* or a *Project Creator* role.
- Been assigned a *Owner* role.

About this task

Every two minutes, the repository is refreshed automatically. You can also manually trigger the refresh activity for a repository.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the left navigation pane or from the **My Projects** page.
3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.

4. In the **Repositories** section, click the **Refresh** icon under **Actions** column for the repository that you want to update.

   The refresh activity is triggered.

   Note: The **Refresh** option is disabled when the refresh is in progress. After the refresh is completed, the **Refresh** option is enabled.

   The following table lists the icons and their description based on the status of the refresh activity:

Refresh icon	Description
	The last refresh activity was successful.
	The last refresh activity was successful but errors were found while processing some of the test assets. To view the error details of the test assets for a specific repository, you can click the arrow inline with the repository name. You can resolve the errors of these test assets in the Git server, and then refresh the server repository, if required. **Note:** The repository, which contains the branch that is selected in the **Branch** field, displays the unprocessed test assets for that branch. Other repositories display the unprocessed test assets for their default branch.
	The refresh activity was not successful due to any one of the following reasons: ◦ Internal error ◦ Connection issue ◦ Authentication failure ◦ Disk quota exceeded ◦ Repository unreachable

Results

You have manually refreshed the repository and the **Last Refreshed** field is updated with the following details:

- The time when it was last refreshed in the remote repository.
- Latest commit ID and message of the remote repository.

Selecting the global branch in a project

After you add one or more repositories to your project on Test , you can start accessing all branches in all repositories of that project. You can select only one branch as a global branch from the list of branches in the **Execution**, **Datasets**, and **Overview** pages.

Before you begin

You must have completed the following tasks:

- Been assigned a *Member* or *Project Creator* role in a team space.
- Added at least one repository in your project.

About this task

You can access all the branches contained in all the repositories, which are added to the project, by using the **Branch** field on the **Execution**, **Datasets**, and **Overview** pages.

After you select a branch from any of the repositories as the global branch, this selection is retained for you as the global branch unless you change the branch on any of the following pages: **Overview**, **Execution**, or **Datasets** pages.

Note:

- When you add the first repository to a project, the global branch is set to the default branch of the repository.
- Branches are arranged in an increasing order of their names in the list.
- When you change the branch in one of the pages, then the same branch is displayed as selected in the other pages and the contents of that page displays the assets from the selected branch.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the left navigation pane or from the **My Projects** page.

   The project **Overview** page is displayed.
3. Select a branch from the **Branch** field.

   The branch that you select on any of the following pages is reflected in the rest of the other pages and the test assets are filtered based on the global branch selection on each of these pages.:
   - **Overview**
   - **Execution**
   - **Datasets**

Results

You have selected a global branch in a project.

Creating webhooks

You can configure a webhook for a repository in the Git server to notify Test whenever there is a push event in the remote repository. Then, the refresh activity is triggered to fetch the latest changes to the server repository.

Before you begin

You must have completed the following tasks:

- Configured the repository that contains the test assets in your project.
- Been assigned a *Member* or *Project Creator* role in a team space.
- Been assigned the *Owner* role in the project.

1. Log in to Test and open the team space that contains your project.
2. Open your project either from the left navigation pane or from the **My Projects** page.
3. Click **Manage > Configuration** from the navigation pane.

   The **Configuration** page of the project is displayed.

4. In the **Repositories** section, click the **Webhook** icon under **Actions** column for the repository that you want to update.

   The **Webhook Configuration** dialog is displayed.

5. Copy the URL and the Secret of the repository that you want to configure.

   Note: The URL and the Secret details are specific to a repository.

6. Click the link of the repository for which you want to configure the webhook.

   The remote repository page is displayed.

7. Paste the details, which you copied from Test, in the appropriate fields of the remote repository.

Results

You have configured the webhook for a repository in the Git server. The changes of the remote repository are synchronized in the repository of your project.

Protecting API test assets by using secrets

Secrets are key-value pairs that are created for your project in Test under a classification. You can create classification for your project that enable you or members in your project to use secrets at test runtime either in Test or in HCL DevOps Test Integrations and APIs (Test Integrations and APIs).

The secrets in a project in Test maintains a separate access control list managed by the members with access to the secrets added to the classification. Controlling access to secrets means controlling access to applications and systems under test. The introduction of secrets under classification for a project simplifies managing access to separate environments. If a

member of a project does not have access to a secret, for example, *a server credential* then the member cannot accidentally or intentionally run tests against that server. For example, tests that must access the database server by using the server credentials to retrieve stored data can only be run by a member if the access to the secrets is granted.

Note: Secrets are applicable to test assets created in Test Integrations and APIs that facilitates running tests in defined environments. Secrets are not applicable to tests created in Test UI or Test Performance.

As a project member with the *Owner* or *Tester* role, you can add secrets to the classification in the project. You can grant or restrict access to the secrets that you added to the classification in the project.

Members with access to a secret classification can access, edit, or delete the secrets in Test and can view secrets, edit secrets, or delete secrets.

Members are granted Read access to the classification at the role level, while Read and Write access can be provided to specific members.

Members in the project with the *Owner* or *Tester* role and with access to the secrets can use the secrets in tests at runtime.

If you are configuring a project to run an API Suite with tests that refer to secret values, you must configure the secrets under a classification by using the **Add Secret** option in the **Security** page. You must complete the following tasks:

1. Create a classification. See Creating a classification on page 934.
2. Add secrets in the classification. See Creating a secret in a classification on page 1750.
3. Grant access to project members or member roles, who can access the secrets. See Granting access to members or member roles on page 1751.

Creating a secret in a classification

You must create secrets in the classifications that is set in your project so that the members of the project with access to the secrets can use them in specific during an API Suite run.

Before you begin

You must have completed the following tasks:

- Created a project on Test.
- Created a classification. See Creating a classification on page 934.
- You must be a member with access to the secrets in the classification. See Granting access to members or member roles on page 1751.

About this task

You can also configure secrets such that the secrets can be used across different test environments by members with access to the classification. Secrets correspond to the environment variables or tags that you create in a Test Integrations and APIs project specific to an environment.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification for which you want to add the secrets under the **Classification**.
5. Click **Add Secret**.
   Result
   The **Add Secret** dialog is displayed.
6. Enter a name for the secret as its *Identifier*, and then enter the `password` as the *Value* for the secret.

   For example, enter the name of the secret to access a database as `dbcred` and enter the `password` required to access the database as its value.
7. Click **Create**.

   A message is displayed that confirms the creation of the secret.

Results

You have created secrets in the selected classification for your project.

What to do next

- You can view, edit, or delete the secrets added to a classification any time you want.
- You can use the secrets in the tests that require these secrets during test runs.

Granting access to members or member roles

You can grant Read access to the secrets added to the classification in your project at the role level for members of a designated role. Additionally, you can provide Read and Write access to specific members. Without access to the classification, members cannot view, create, edit, delete, or use the secrets added to the classification.

Before you begin

You must be a project **Owner** and have completed the following tasks:

- Created a project in Test. See Managing DevOps Test projects on page 1724.
- Added one or more users to your project. See Adding users to a project on page 1732.
- Created at least one classification. See Creating a classification on page 934.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

The **Overview** page of the project is displayed.

3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Click a classification in the list under **Classification** to which the secrets are added and for which you want to grant the access.
5. Select any of the following methods to grant access to a member:
   Choose from:

   - Select the checkbox for the specific role under **Grant Read Only Access to roles** to grant read only access to the members of a specific role. For example, if you select **Tester**, then all members in the project with a tester role are granted read only access to the selected classification. You can select any role or all the roles listed.
   - Perform the following steps to grant read/write access to specific members:
     a. Click **Add Members** inline with **Members with Read/Write Access**.

        The **Additional Members** dialog is displayed.

     b. Enter the name or email ID to search the user or navigate through the list and select the member you want to grant access. You can select multiple members from the list.

        Important:

        Irrespective of the role that the member (**Owner**, **Tester** or **Viewer**) was assigned in the project, the access to the classification to which the secret is added must be specifically granted to the members from the **Security** page.

     c. Click **Add Members.**

        You granted the read/write access to the added members.
6. Select any of the following methods to revoke access to a member:
   Choose from:

   - Clear the checkbox for the specific role under **Grant Read Only Access to roles,** to revoke read only access to the members of a specific role. For example, if you clear the **Tester** checkbox then access for all members in the project with a tester role is revoked for the selected classification. You can select any role or all the roles listed.
   - Click the **Delete** icon inline with the member name to remove the specific members from the **Members with Read/Write Access** list.

Notes:

- Any member with access to the classification can remove access to other members specifically added or of all members with a specific role.
- Members with access to the classification can remove themselves from the access list provided that there is at least one member in the list. After removing themselves, members cannot add themselves back to the access list and must be added by any of the other members in the list.

Results

You have added members from your project or members with specific role to the access list of people who can access secrets in the selected classification, or you have removed specific members or members with specific role from the access list.

What to do next

You can add secrets to the classifications for your project.

Managing secrets

You can view, edit, or delete the secrets configured under a classification any time after you created secrets or after you were granted access to the secrets classification. You can change the value of the secret by editing the secret. You can delete secrets that you no longer require in your test environment.

Before you begin

You must have completed the following tasks:

- You must have created a project on Test and configured a secrets in your project.
- You must have created secrets in the selected classification or the classifications must contain secrets.
- You must be a member with access to the secrets classification.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Security**.
   The **Security** page is displayed.
4. Perform the following steps:
   a. Select the **Classification** from the list in the **Security** page.
   b. Optionally, select the classification that you want from the list if there are multiple classifications in the project.

   The secrets configured in the selected classifications are displayed.
5. Complete the steps for the task you want to perform as listed in the following table:

Task	Action
Viewing a secret value	Click the **View** icon inline with the secret to view the entered value. The value most likely is a password for the secret. The value configured for the secret is displayed.
Editing a secret value	Perform the following steps: a. Click the **Edit** icon inline with secret that you want to edit, The **Edit secret** dialog is displayed. b. Enter a `new value` for the secret as its *Value*. The value can be a password for the secret. **Note:** You can only change the value of the secret. c. Click **Save**. The value of the selected secret is changed.
Deleting a secret	Perform the following steps: a. Click the **Delete** icon inline with the secret that you want to delete. The **Delete secret** dialog is displayed. b. Click **Delete**. After you delete the secrets, the secrets listed in the classification is removed from the list.

Results

- You viewed the password configured of the secret under a classification that you created or were granted access.
- You changed the secret value of the secret under a classification in your project.
- You deleted and removed the secret from the selected classification in your project.

What to do next

You can use secrets in the tests that require these secrets during test runs.

Related information

Becoming a project member on page 1735

Test run configurations on page 1134

Notification Management

Test provides a feature to display notifications for different events within the user interface (UI) of Test. You can configure Email notifications on Test to send out notifications about different events as emails to the subscribed users. You can also configure third-party applications as webhooks on Test when you want Test to send push notifications about the server events to the third-party applications in a preset template.

Test provides you the following features to receive notifications about server events:

Channels	Description
In-app	Notifications are displayed within the UI of Test.
Email	Notifications are sent in emails to subscribed users of Test by an SMTP server.
Webhooks	Notifications in a preset template are sent as push notifications by Test to the configured webhook of a third-party application, such as MS Teams, Slack, or others.

Managing notifications in the server UI

Viewing the in-app notifications or notifications in the server UI is a feature that is enabled automatically when you register as a user in Test.

When you register as a user in Test, the in-app notifications are automatically subscribed for you.

If you want to receive notifications about specific events and not all events, you can change the default subscriptions and modify your subscription for the server events. You can also view the in-app notifications that are displayed for you. See Managing in-app notifications on page 1756.

Managing Email notifications

You, as a *Server Administrator* or *Team Space Owner* can configure an existing Simple Mail Transfer Protocol (SMTP) server so that Test can send out notifications about the server events in emails to the registered users who have subscribed to the email notifications. See Configuring an SMTP server to manage email notifications on page 1761.

If you, as a registered user of Test want to receive the email notifications about specific events and not all events, you can change the subscriptions for the server events. See Managing email notifications on page 1759.

Managing webhook notifications

You, as a *Server Administrator*, *Team Space Owner*, or *Project Owner* can configure third-party applications as webhooks so that Test can send out push notifications about the server events to the third-party applications. See Configuration of a webhook on page 1762.

Managing in-app notifications

You can manage the in-app notifications by changing the subscriptions that are enabled by default for all the events that occur on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.

About this task

You can view the default notification subscription for all server events, and then choose to unsubscribe or change the subscription for any or all the events.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **User** icon on the menu bar.
3. Click **Account Settings**.

   Alternatively, click the **Notification** icon on the menu bar, and then click the **Settings** icon in the **Notifications** dialog.
4. Click the Expand icon and identify the server events for which you want to change the subscription.
   You can notice that for each of the server events, the in-app channel is displayed as selected, which is the default selection unless you have modified the subscription.
5. Perform the actions indicated in the following table to clear the subscriptions for server events that you do not want to be notified and to select the in-app subscription for the server events that you want to be notified.

When	Action	Result
You do not want to receive the in-app	Perform the following steps:	You have unsubscribed the in-app notifications for a specific server event and

When	Action	Result
notifications for a specific server event.	a. Identify the server event by expanding the channel list. b. Clear the checkbox in the **In App** row to unsubscribe. c. Click **Apply**.	the notification is not displayed for you when the event occurs.
You want to receive the in-app notifications for a specific server event.	Perform the following steps: a. Identify the server event by expanding the channel list. b. Select the checkbox in the **In App** row. c. Click **Apply**.	You have subscribed to the in-app notifications for the server event and the notification is displayed for you when the event occurs.
You want to subscribe to the in-app notifications for all server events.	Select the **In App** checkbox under **Channels**, and then click **Apply**.	The in-app channel is displayed as selected for all server events and notifications for all server events will be displayed.
You do not want to receive any in-app notifications for all server events.	Clear the **In App** checkbox, and then click **Apply**.	The in-app channel selections for all server events are cleared and no notifications will be displayed when the event occurs.

6. To change the notification language in which you want to receive the notifications from the default, select a language from the **Email Language** option in the **Notifications** pane and click **Apply**.

Results

You have modified the notification subscription for specific or all server events.

Related information

Managing email notifications on page 1759

Viewing in-app notifications

You can view the notifications for the events that occur within the Test .

About this task

The in-app notifications are displayed in the server UI under the following categories for the server events that occur:

Category	Notifications displayed
User	Include information about actions performed by team space members in the team space.
System	Include information about actions performed by registered users at the server level.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Notifications** icon on the menu bar.

   The **Notifications** dialog is displayed.

   You can find the information about the default notifications that you can view based on your role in the server as in-app notifications in the **USER** tab and **SYSTEM** tab.

Category	Notification details	*Server Administrator* or *Team Space Owner*	*Project Creator* or *Project Owner*	*Tester*	*Viewer*
User notifications	Information about events at the server level or across team spaces performed by other team space owners.	✔	✔	✘	✘
	Information about events in the team space performed by other team space members.	✔	✘	✘	✘
	Information about events in the project performed by other project members.	✘	✔	✔	✘
	Information about events performed by you in a project.	✘	✘	✔	✔
System notifications	Information about events at the server level or across team spaces.	✔	✔	✘	✘

Category	Notification details	*Server Administrator* or *Team Space Owner*	*Project Creator* or *Project Owner*	*Tester*	*Viewer*
	Information about events in the team space performed by other team space members.	✔	✔	✘	✘
	Information about events in the projects at the team space level performed by other project members.	✔	✔	✘	✘

The notifications in the **USER** tab are displayed as the default option.

3. View the notifications in the **USER** tab.
4. View the notifications at the server level by clicking the **SYSTEM** tab.

Results

You have viewed the in-app notifications.

Managing email notifications

You can manage the email notifications by changing the subscriptions that are automatically enabled for you for all events that occur in Test .

Before you begin

You must have completed the following tasks:

- Ensured that the *Server Administrator* or *Team Space Owner* has configured a Simple Mail Transfer Protocol (SMTP) server. See Configuring an SMTP server to manage email notifications on page 1761.
- Ensured that you are assigned a role as a *Member* or *Project Creator* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you are assigned a role as a *Viewer*, *Tester*, or *Project Owner* in the project.

About this task

After the SMTP server is configured on Test, you can view the server events that are subscribed in the **General Settings** page. You can either unsubscribe from the **Email** channel or change your subscriptions of the server events for which you do not want to receive email notifications.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **User** icon on the menu bar.
3. Click **Account Settings**.

   Alternatively, click the **Notification** icon on the menu bar, and then click the **Settings** icon in the **Notifications** dialog.

   The **General Settings** dialog is displayed.
4. Perform the actions indicated in the following table for the option you want for the email notifications:

When	Action	Results
You want to subscribe to email notifications for all server events.	Select the **Email** checkbox under **Channels** then click **Apply**.	The email channel for all server events is selected and you are subscribed to receive email notifications for all server events.
You do not want to receive any email notifications.	Uncheck the **Email** checkbox under **Channels**, then click **Apply**.	The email notification selection for all server events is cleared and you are unsubscribed from receiving email notifications for all the server events.
You want to receive temail notifications for a specific server event.	Perform the following steps: a. Expand the channel list in the row of the server event. b. Select the **Email** checkbox of the event you want to subscribe. c. Click **Apply**.	You have subscribed to the email channel to receive email notifications for the server event.
You do not want to receive email notifications for a specific server event, if you subscribed to it previously.	Perform the following steps: a. Expand the channel list in the row of the server event. b. Clear the checkbox for the **Email** channel option of the event you want to unsubscribe. c. Click **Apply**.	You have unsubscribed from the email channel for the server event, and you will not receive email notifications when the event occurs.

5. To change the language in which you want to receive the email notifications from the default, select a language from the **Email Language** list in the **Notifications** pane and click **Apply**.

   The default language is the language set on the computer that hosts Test.

Results

You completed the following tasks:

- Subscribed to the email channel for receiving email notifications for the selected server events.
- Unsubscribed from the email channel for selected server events that you subscribed previously.
- Subscribed to the email channel for receiving email notifications for all server events.
- Unsubscribed from the email channel for receiving email notifications for all server events.

Related information

Managing in-app notifications on page 1756

Configuring an SMTP server to manage email notifications

Before you can receive email notifications about events that occur in Test , you need to configure an existing Simple Mail Transfer Protocol (SMTP) server in Test. Once configured, email notifications are sent to the registered users if they are subscribed to email notifications whenever events occur in Test.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that an existing SMTP server is available to be connected to Test.

About this task

The configuration works for emails that are supported by the SMTP server. To use this capability, you must enable Test to connect to the SMTP server to send email notifications.

As a server administrator, you can configure email notifications so that other users can receive information about actions on the server projects.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Server administration page** icon from the menu bar.

   The **Global Settings** page is displayed.
3. Click **Add Configuration**.

4. Provide the following details to configure the SMTP server:
    - A name to the configuration.
    - The hostname and port number of the SMTP server.

      Note: The default configuration uses a local SMTP server that listens to the default port 25. This SMTP server does not require any authentication.

    - The email address of the sender. The email address is displayed in email notifications. The email notifications are sent to the users based on the notification settings.
    - The encryption type for secure communications is when the SMTP server is on an external network. You can enable either **STARTTLS** or **SSL/TLS**.

      Note: The SMTP server for the secure communication listens to the default port 465.

    - Login credentials are required if the SMTP server requires authentication.
5. Click **Add**.

Note: The SMTP configuration is added after a valid connection is established with the SMTP server.

Results

You configured an SMTP server to connect with Test and send out email notifications about the server events to users who have subscribed to email notifications.

What to do next

You can modify the default subscriptions for the email notifications, see Managing email notifications on page 1759.

Related information

Notification Management on page 1755

Configuration of a webhook

You can configure webhooks to send notifications to a messaging application such as Microsoft Teams, Slack, or any generic application that can receive an HTTP POST request when any of the events occur on Test . For example, when you add or remove a member from a team space or when you run a test.

A webhook is an HTTP custom callback that provides an application to send notifications to other applications. Webhooks help applications automatically communicate with other external applications.

The following table lists the tasks that you can perform and manage for the webhooks:

Task	Go to...
Configure a webhook template	Configuring a webhook template on page 1763
Configure a webhook	Configuring a project webhook on page 1774
Edit, clone, or delete a webhook template	Managing the webhook template configuration on page 1776
Edit, clone, or delete a webhook	Managing the webhook configuration on page 1779

Webhooks overview

Webhooks send a notification to the configured messaging applications when specific events occur on Test .

When the event occurs, Test automatically sends an HTTP POST request with event-related information in a JSON format or in plain text to the URL that you configured. You can define a template body or use the sample template that is set on Test. You must create a channel type, a template, and a webhook to send an HTTP POST request.

You can manage the webhook template configuration and webhook configuration on Test. You can identify the events and view the status of the events that are posted on the messaging application whenever an event occurs.

Configuring a webhook template

You can configure a project, team space and a server webhook template on Test to send notifications to the configured messaging application.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Ensured that you are familiar with Apache Velocity. See Apache Velocity Documentation.

About this task

You can create a template based on the channel type that you configured. You can select a generic template to send notifications to the generic application. You can also select a JSON sample template with specific fields defined to display event-related notifications in Microsoft Teams or Slack. You can use the sample template that is available for the messaging application or you can also define a JSON template to post the message about the events that occur on Test.

1. Select from the following actions you want to perform:
   - Configure a project webhook template. Go to Step 2 on page 1764.
   - Configure a team space webhook template. Go to Step 3 on page 1766.
   - Configure a server webhook template. Go to Step 4 on page 1768.
2. Perform the following steps to configure a project webhook template.

   a. Log in to Test.

      The **Projects** page of the initial team space is displayed.

   b. Click **My projects > *project_name*** to open the project that contains the test assets.

      The **Overview** page of the project is displayed.

   c. Click **Manage > Webhooks**.

      The **Templates** tab of Webhooks is displayed.

   d. Click **New Template**.

      The **New template** page is displayed.

   e. Perform the following actions on the **DETAILS** tab:
      i. Enter a unique name for the template.
      ii. Click the **Channel Type** option, and then select an existing channel type.
      iii. Alternatively, you can create a new channel type by clicking **Create new channel type ...** option if there are no channel types available in the drop-down list.

         The **New channel type** dialog is displayed.

      iv. Enter a unique name for the channel type, and then enter a description for the channel type.

      Note: You can also create a new channel type or edit a channel type from the **Channel Type** option available within the **Templates** tab.

   f. Perform the following steps if you want to select either a channel icon or a brand icon:
      i. Click **Create new channel type...**, and then click **Channel type** icon .

         The **Select icon** page is displayed.

      ii. View a list of icons that are displayed In the **GENERAL** tab.
      iii. Search for the icons by entering any text contained in the icon name in the **Search** field, and then select the required icon.

         The selected icon is displayed in the **Selected icon** box.

      iv. Click the **BRANDS** tab.

         View a list of icons that are displayed.

v. Search for the brands by entering any text contained in the brand name in the **Search** field, and then select the brand.

The selected brand is displayed in the **Selected icon** box.

vi. Click **Ok**.

The default icon is replaced with the selected icon.

g. Enter a description for the channel type, and then click **Apply**.

The channel type is created and added to the **Channel Type** list.

h. Enter a description for the template.

i. Perform the following steps to select an event:

i. Click **This template is suitable for events** and select All Events, or any of the subset events from the following list:

- All Events
- Team Space Events
- Project Events
- Execution Events
- License Events

ii. Search for the events by entering any text contained in the events name in the **Search** field, and then select the corresponding checkbox to select the specific event.

Note: All Events is the default event that is set for the webhook template.

j. Click the **Template** tab, and then perform any of the following actions:

- Perform the following steps to insert a generic template:

i. Click the **cloud** icon to insert a velocity text processing template in the **Template body**.

A message is displayed about the action that inserts a text in the template body.

ii. Click **Ok**.

The text is inserted in the **Template body**.

- Perform the following steps to insert a Microsoft Teams sample template:

i. Click the **Microsoft Teams** icon to replace a template with a Microsoft Teams template.

A message is displayed about the action that replaces the content of the template body with a Microsoft Teams template.

ii. Click **Ok**.

A Microsoft Teams sample template is inserted in the **Template body**.

- Perform the following steps to insert a Slack sample template:

i. Click the **Slack** icon .
ii. A message is displayed about the action that replaces the content of the template body with a Slack template.
iii. Click **Ok**.

A Slack sample template is inserted in the **Template body.**

- Perform the following steps to insert a DevOps Velocity sample template:

i. Click the **infinity** icon .
ii. A message is displayed about the action that replaces the content of the template body with a DevOps Velocity template.
iii. Click **Ok**.

A DevOps Velocity sample template is inserted in the **Template body.**

k. Click **Apply**.

The template is created and added to the **Templates** tab.

3. Perform the following steps to configure a team space webhook template:

a. Log in to Test.

The **Projects** page of the initial team space is displayed.

b. Click **Initial Team Space > Manage Team Space > > Webhooks**.

The **Templates** tab of Webhooks is displayed.

c. Click **New Template**.

The **New template** page is displayed.

d. Perform the following actions on the **DETAILS** tab:
i. Enter a unique name for the template.
ii. Click the **Channel Type** option, and then select an existing channel type.
iii. Alternatively, you can create a new channel type by clicking **Create new channel type ...** option if there are no channel types available in the drop-down list.

The **New channel type** dialog is displayed.

iv. Enter a unique name for the channel type, and then enter a description for the channel type.

Note: You can also create a new channel type or edit a channel type from the **Channel Type** option available within the **Templates** tab.

e. Perform the following steps if you want to select either a channel icon or a brand icon:

i. Click **Create new channel type...**, and then click **Channel type** icon .

The **Select icon** page is displayed.

ii. View a list of icons that are displayed In the **GENERAL** tab.

iii. Search for the icons by entering any text contained in the icon name in the **Search** field, and then select the required icon.

The selected icon is displayed in the **Selected icon** box.

iv. Click the **BRANDS** tab.

View a list of icons that are displayed.

v. Search for the brands by entering any text contained in the brand name in the **Search** field, and then select the brand.

The selected brand is displayed in the **Selected icon** box.

vi. Click **Ok**.

The default icon is replaced with the selected icon.

f. Enter a description for the channel type, and then click **Apply**.

The channel type is created and added to the **Channel Type** list.

g. Enter a description for the template.

h. Perform the following steps to select an event:

i. Click **This template is suitable for events** and select All Events, or any of the subset events from the following list:

- All Events
- Team Space Events
- Project Events
- Execution Events
- License Events

ii. Search for the events by entering any text contained in the events name in the **Search** field, and then select the corresponding checkbox to select the specific event.

Note: All Events is the default event that is set for the webhook template.

i. Click the **Template** tab, and then perform any of the following actions:

- Perform the following steps to insert a generic template:

i. Click the **cloud** icon to insert a velocity text processing template in the **Template body**.

A message is displayed about the action that inserts a text in the template body.

ii. Click **Ok**.

The text is inserted in the **Template body**.

- Perform the following steps to insert a Microsoft Teams sample template:
  i. Click the **Microsoft Teams** icon to replace a template with a Microsoft Teams template.

  A message is displayed about the action that replaces the content of the template body with a Microsoft Teams template.

  ii. Click **Ok**.

  A Microsoft Teams sample template is inserted in the **Template body**.

- Perform the following steps to insert a Slack sample template:
  i. Click the **Slack** icon .
  ii. A message is displayed about the action that replaces the content of the template body with a Slack template.
  iii. Click **Ok**.

  A Slack sample template is inserted in the **Template body.**

- Perform the following steps to insert a DevOps Velocity sample template:
  i. Click the **infinity** icon .
  ii. A message is displayed about the action that replaces the content of the template body with a DevOps Velocity template.
  iii. Click **Ok**.

  A DevOps Velocity sample template is inserted in the **Template body.**

j. Click **Apply**.

The template is created and added to the **Templates** tab.

4. Perform the following steps to configure a server webhook template:

a. Log in to Test.

The **Projects** page of the initial team space is displayed.

b. Click the **Settings** icon on the menu bar.
c. Click **Webhooks**.

The **Templates** tab of Webhooks is displayed.

d. Click **New Template**.

The **New template** page is displayed.

e. Perform the following actions on the **DETAILS** tab:

i. Enter a unique name for the template.
ii. Click the **Channel Type** option, and then select an existing channel type.
iii. Alternatively, you can create a new channel type by clicking **Create new channel type ...** option if there are no channel types available in the drop-down list.

The **New channel type** dialog is displayed.

iv. Enter a unique name for the channel type, and then enter a description for the channel type.

Note: You can also create a new channel type or edit a channel type from the **Channel Type** option available within the **Templates** tab.

f. Perform the following steps if you want to select either a channel icon or a brand icon:

i. Click **Create new channel type...**, and then click **Channel type** icon .

The **Select icon** page is displayed.

ii. View a list of icons that are displayed In the **GENERAL** tab.
iii. Search for the icons by entering any text contained in the icon name in the **Search** field, and then select the required icon.

The selected icon is displayed in the **Selected icon** box.

iv. Click the **BRANDS** tab.

View a list of icons that are displayed.

v. Search for the brands by entering any text contained in the brand name in the **Search** field, and then select the brand.

The selected brand is displayed in the **Selected icon** box.

vi. Click **Ok**.

The default icon is replaced with the selected icon.

g. Enter a description for the channel type, and then click **Apply**.

The channel type is created and added to the **Channel Type** list.

h. Perform the following steps to select an event:

i. Click **This template is suitable for events** and select All Events, or any of the subset events from the following list:

- All Events
- Team Space Events
- Project Events
- Execution Events
- License Events

ii. Search for the events by entering any text contained in the events name in the **Search** field, and then select the corresponding checkbox to select the specific event.

Note: All Events is the default event that is set for the webhook template.

i. Click the **Template** tab, and then perform any of the following actions:

- Perform the following steps to insert a generic template:
  i. Click the **cloud** icon to insert a velocity text processing template in the **Template body**.

  A message is displayed about the action that inserts a text in the template body.

  ii. Click **Ok**.

  The text is inserted in the **Template body**.
- Perform the following steps to insert a Microsoft Teams sample template:
  i. Click the **Microsoft Teams** icon to replace a template with a Microsoft Teams template.

  A message is displayed about the action that replaces the content of the template body with a Microsoft Teams template.

  ii. Click **Ok**.

  A Microsoft Teams sample template is inserted in the **Template body**.
- Perform the following steps to insert a Slack sample template:
  i. Click the **Slack** icon .
  ii. A message is displayed about the action that replaces the content of the template body with a Slack template.
  iii. Click **Ok**.

  A Slack sample template is inserted in the **Template body.**
- Perform the following steps to insert a DevOps Velocity sample template:
  i. Click the **infinity** icon .
  ii. A message is displayed about the action that replaces the content of the template body with a DevOps Velocity template.
  iii. Click **Ok**.

  A DevOps Velocity sample template is inserted in the **Template body.**

j. Click **Apply**.

The template is created and added to the **Templates** tab.

Results

You have configured and created a project, team space and a server webhook template on Test.

What to do next

You must configure a webhook to push the events from Test to the messaging application. See Configuring a project webhook on page 1774.

Configuring a server webhook

You can configure a server webhook to send notifications to a messaging application whenever an event occurs on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role of a *Server Administrator*. See Managing access to the server projects on page 1735.
- Ensured that you have access to the messaging applications.
- Created a channel type.
- Configured your messaging application.
- Copied the URL of the external communication application. See Creating an incoming webhook.
- Created a webhook template. See Configuring a webhook template on page 1763.

About this task

When you configure a webhook on Test, you must copy the URL of the communication application in the **Webhook URL** field and paste the outgoing URL, and then select the events. You can select either all events or a subset of events.

You can be notified of the following types of events:

- All Events
- Team space Events
- Project Events
- Execution Events
- License Events

You can configure the webhook resume policy by selecting either the Queue old messages or Discard old messages option. When a server webhook is disabled due to communication failures or network issues, the server event notifications are not delivered to the configured webhook, and you receive a notification that the webhook is disabled. By configuring the queue old messages, you can ensure that the event notifications are not lost and are delivered to the configured webhook after the server webhook is enabled again.

If you configure the Discard old messages for a server webhook, the event notifications are discarded from the moment the webhook is disabled. Only server events that occur after the webhook is re-enabled are sent to the configured webhook.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click the **Settings** icon on the menu bar, and then click **Webhooks**.
   The **Templates** tab of Webhooks is displayed.
3. Click the **Webhooks** tab, and then click **New Webhook**.

   The **New Webhook** page is displayed.
4. Enter a name for the webhook and paste the URL that you copied in the **Webhook URL** field.
   You must copy the URL while you configure your communication application.
5. Click the **Webhook resume policy**, and then select a resume policy option from the drop-down list. You can select any of the following options:
   a. Queue old messages
   b. Discard old messages
6. Click the **Channel Type**, and then select a channel type from the list.
7. Enter a description for the webhook, and then select a template from the drop-down list.
   A corresponding event that you configured for the template is displayed in the **Events** field.
8. Click the **Templates** icon Template, to add the templates that are configured for the channel type.

   Note: If you want to change the events, then click the events option and select the required events.
9. You can perform any of the following steps:
   a. Click move the template to a higher priority icon to move the template to a higher priority.
   b. Click move the template to a lower priority icon to move the template to a lower priority.
10. Click **Apply**.
    The webhook is created and added to the **Webhooks** tab.

Results

You have configured and created a server webhook on Test.

What to do next

You can perform any of the following tasks:

- You can duplicate or delete the template that you created on Test. See Managing the webhook template configuration on page 1776.
- You can duplicate or delete the webhook you created on Test. See Managing the webhook configuration on page 1779.

Configuring a team space webhook

You can configure a team space webhook to send notifications to a messaging application whenever an event occurs on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Ensured that you have access to the messaging applications.
- Created a channel type.
- Configured your messaging application.
- Copied the URL of the external communication application. See Creating an incoming webhook.
- Created a webhook template. See Configuring a webhook template on page 1763.

About this task

When you configure a webhook on Test, you must copy the URL of the communication application in the **Webhook URL** field and paste the outgoing URL, and then select the events. You can select either all events or a subset of events.

You can be notified of the following types of events:

- All Events
- Team space Events
- Project Events
- Execution Events
- License Events

You can configure the webhook resume policy by selecting either the Queue old messages or Discard old messages option. When a team space webhook is disabled due to communication failures or network issues, the team space event notifications are not delivered to the configured webhook, and you receive a notification that the webhook is disabled. By configuring the queue old messages, you can ensure that the event notifications are not lost and are delivered to the configured webhook after the team space webhook is enabled again.

If you configure the Discard old messages for a team space webhook, the event notifications are discarded from the moment the webhook is disabled. Only team space events that occur after the webhook is re-enabled are sent to the configured webhook.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **Initial Team Space > Manage team space**, and then click **Webhooks**.
   The **Templates** tab of Webhooks is displayed.
3. Click the **Webhooks** tab, and then click **New Webhook**.

   The **New Webhook** page is displayed.
4. Enter a name for the webhook and paste the URL that you copied in the **Webhook URL** field.
   You must copy the URL while you configure your communication application.

5. Click the **Webhook resume policy**, and then select a resume policy option from the drop-down list. You can select any of the following options:
    a. Queue old messages
    b. Discard old messages
6. Click the **Channel Type**, and then select a channel type from the list.
7. Enter a description for the webhook, and then select a template from the drop-down list.
   A corresponding event that you configured for the template is displayed in the **Events** field.
8. Click the **Templates** icon , to add the templates that are configured for the channel type.

   Note: If you want to change the events, then click the events option and select the required events.

9. You can perform any of the following steps:
    a. Click move the template to a higher priority icon to move the template to a higher priority.
    b. Click move the template to a lower priority icon to move the template to a lower priority.
10. Click **Apply**.
    The webhook is created and added to the **Webhooks** tab.

Results

You have configured and created a team space webhook on Test.

What to do next

- You can duplicate or delete the template that you created on Test. See Managing the webhook template configuration on page 1776.
- You can duplicate or delete the webhook you created on Test. See Managing the webhook configuration on page 1779.

Configuring a project webhook

You can configure a project webhook to send notifications to a messaging application whenever an event occurs on Test .

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Created a project in your team space in Test. See Creating a project on page 1728.
- Ensured that you have access to the messaging applications.
- Created a channel type.
- Configured your messaging application.

- Copied the URL of the external communication application. See Creating an incoming webhook.
- Created a webhook template. See Configuring a webhook template on page 1763.

About this task

When you configure a webhook on Test, you must copy the URL of the communication application in the **Webhook URL** field and paste the outgoing URL, and then select the events. You can select either all events or a subset of events.

You can be notified of the following types of events:

- All Events
- Team space Events
- Project Events
- Execution Events
- License Events

  You can configure the webhook resume policy by selecting either the Queue old messages or Discard old messages option. When a project webhook is disabled due to communication failures or network issues, the project event notifications are not delivered to the configured webhook, and you receive a notification that the webhook is disabled. By configuring the queue old messages, you can ensure that the event notifications are not lost and are delivered to the configured webhook after the project webhook is enabled again.

  If you configure the Discard old messages for a project webhook, the event notifications are discarded from the moment the webhook is disabled. Only project events that occur after the webhook is re-enabled are sent to the configured webhook.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Webhooks** in the navigation pane.
   The **Templates** tab of Webhooks is displayed.
4. Click the **Webhooks** tab, and then click **New Webhook**.

   The **New Webhook** page is displayed.
5. Enter a name for the webhook and paste the URL that you copied in the **Webhook URL** field.
   You must copy the URL while you configure your communication application.
6. Click the **Webhook resume policy**, and then select a resume policy option from the drop-down list. You can select any of the following options:
   a. Queue old messages
   b. Discard old messages

      The Queue old messages is selected as the default option.

7. Click the **Channel Type**, and then select a channel type from the list.
8. Enter a description for the webhook, and then select a template from the drop-down list.
   A corresponding event that you configured for the template is displayed in the **Events** field.
9. Click the **Templates** icon, to add the templates that are configured for the channel type.

   Note: If you want to change the events, then click the events option and select the required events.

10. You can perform any of the following steps:
    a. Click move the template to a higher priority icon to move the template to a higher priority.
    b. Click move the template to a lower priority icon to move the template to a lower priority.
11. Click **Apply**.
    The webhook is created and added to the **Webhooks** tab.

Results

You have configured and created a project webhook on Test.

What to do next

You can perform any of the following tasks:

- You can duplicate or delete the template that you created on Test. See Managing the webhook template configuration on page 1776.
- You can duplicate or delete the webhook you created on Test. See Managing the webhook configuration on page 1779.

Managing the webhook template configuration

You can manage the webhook template configuration that you set on Test when you want to edit, duplicate, delete the existing webhook templates, or create new templates. Before you create a webhook, you can view the templates created at server, team space, and project levels. Additionally when you create a project, team space or server webhook you can utilize the corresponding templates associated with them.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Project Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 1721.
- Configured a webhook template. See Configuring a webhook template on page 1763.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Webhooks** in the navigation pane.

   The **Templates** tab of Webhooks is displayed.
4. Perform the tasks indicated in the following table:

Task	Action
Configuring a webhook template	Click the **New Template** option to configure a webhook template. See Configuring a webhook template on page 1763.
Viewing webhook templates	View the following details about the project, team space and server templates: a. The **Show all levels** checkbox selected. b. The following icons that display the server, team space, and project webhook templates in the **From** column. ▪ Project template icon ▪ Team space template icon ▪ Serve template icon As a project owner, you can view project, team space, and server templates on the project template page. Similarly, as a team space owner, you can view only team space and server templates on the team space template page. Perform the following steps to view project or team space templates: a. Clear the **Show all levels** checkbox. You can only view project webhook templates, if you are on the project template page, and you can only view team space templates, if you are on the team space template page. b. Click the **Action** icons to edit, duplicate view or delete the templates. **Note:** As a *Project Owner*, you can edit the project webhook templates; however, you can only view and clone the details of server webhook templates or team space webhook templates. If you are a *Team Space*

Task	Action
	Owner you can edit the team space templates; however, you can only view and clone the details of server webhook templates. If you clone a webhook template, then you can edit or delete a webhook template.
Editing a webhook template	Perform the following steps to edit the webhook template: a. Click the **Edit template** icon . The **Edit template** page is displayed. b. In the **Details** tab, you can enter a new name for the webhook template, or select any other value for the options. c. In the **Templates** tab, you can also edit or select a template based on the channel type you selected on **Details** tab. d. Click **Apply**. The edited template is saved to the **Templates** tab.
Cloning a webhook template	Perform the following steps to duplicate the webhook template: a. Click the **Duplicate template** icon . The **Duplicate template** page is displayed. b. In the **Details** tab, you can edit or modify the values by entering a new name for the webhook template, by selecting a Channel Type, or Events. c. In the **Templates** tab, you can edit or select other webhook template based on the channel type you selected on **Details** tab. d. Click **Apply**. The cloned webhook template is added to the **Templates** tab.
Deleting a webhook template	Perform the following steps to delete the webhook template: a. Click the **Delete Template** icon . The Delete template confirmation dialog is displayed. b. Select the **I understand template deletion is permanent** checkbox, and then click **Delete**. The webhook template is removed from the template list.

Results

You achieved the following results:

- Edited and saved a webhook template configuration.
- Cloned a webhook template configuration.
- Deleted a webhook template configuration.

Managing the webhook configuration

You can manage the webhook configuration that you set on Test you want to edit, duplicate, delete the existing webhook configurations, or create a webhook.

Before you begin

- Ensured that you are assigned a role of a project *Owner* in the project. See Managing access to the server projects on page 1735.
- Ensured that you configured a webhook. See Configuring a project webhook on page 1774.

1. Log in to Test.
   The **Projects** page of the initial team space is displayed.
2. Click **My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
3. Click **Manage > Webhooks** in the navigation pane.
   The **Templates** tab of Webhooks is displayed.
4. Click the **Webhooks** tab.
5. Perform the tasks indicated in the following table:

Task	Action
New Webhook	Click the **New Webhook** option to configure and create a webhook. See Configuring a project webhook on page 1774.
Editing a webhook	Perform the following steps to edit the webhook: a. Click the **Edit Webhook** icon . The **Edit webhook** page is displayed. b. Enter a new name for the webhook. c. Replace the webhook URL by pasting a different URL of another messaging channel you configured. d. Select another channel type to post the notification of an event. e. Select a different template, and then select events from the **Events** list. f. Click **Close**.
Cloning a webhook	Perform the following steps to clone the webhook: a. Click the **Duplicate webhook** icon . The **Duplicate webhook** page is displayed.

Task	Action
	b. Enter a new name for the webhook, and then click **Apply**. The cloned webhook is added to the **Webhooks** tab.
Deleting a webhook	Perform the following steps to delete the webhook: a. Click the **Delete Webhook** icon . The **Delete Webhook** confirmation dialog is displayed. b. Select the **I understand webhook deletion is permanent** checkbox, and then click **Delete**. The webhook is removed from the webhook list.

Results

You achieved the following results:

- Edited and saved a webhook configuration.
- Cloned a webhook configuration.
- Deleted a webhook configuration.

Enabling a webhook

You can enable a webhook when push notifications about events occurring on Test are not delivered to the configured webhook due to communication failures or network issues, and you receive a notification that the webhook is disabled.

Before you begin

You must have completed the following tasks:

- Ensured that you are assigned the role of a *Server Administrator*, *Team Space Owner* or a *Project Owner*. If you are a server administrator, you can have three roles-server admin, team space owner, and project owner.
- Configured a webhook. See Configuring a project webhook on page 1774.

About this task

When a webhook is disabled due to communication failure or network issues, you are notified about the disablement of the webhook. Consequently notifications for the events are not delivered to the configured webhook and instead be queued for sending after you re-enable the webhook. You have the option to either send the queued messages or discard the queued messages.

1. Select from the following actions that you want to perform:
   - Enable a project webhook. Go to Step 2 on page 1781.
   - Enable a team space webhook. Go to Step 3 on page 1781.
   - Enable a server webhook. Go to Step 4. on page 1781
2. Perform the following steps to manually enable a project webhook:

   a. Log in to Test.

      The **Projects** page of the initial team space is displayed.

   b. Click **My projects >** ***project_name*** to open the project that contains the test assets.

      The **Overview** page of the project is displayed.

   c. Click **Manage > Webhooks**.

      The **Templates** tab of Webhooks is displayed.

   d. Click the **Webhooks** tab, and then click the **Disabled** webhook option to enable a webhook that is disabled.

   The event notifications that were queued when the webhook was disabled are delivered to the configured webhook or discarded by Test.This setting is based on the resume policy configuration that you set for the specific project webhook.
3. Perform the following steps to manually enable a team space webhook:

   a. Log in to Test.

      The **Projects** page of the initial team space is displayed.

   b. Click the **Settings** icon on the navigation pane, and then click **Manage team space**.
   c. Click **Webhooks**.

      The **Templates** tab of Webhooks is displayed.

   d. Click the **Webhooks** tab, and then click the **Disabled** webhook option to enable a webhook that is disabled.

   The event notifications that were queued when the webhook was disabled are delivered to the configured webhook or discarded by Test. This setting is based on the resume policy configuration that you set for the team space webhook.
4. Perform the following steps to manually enable a server webhook:

   a. Log in to Test.

      The **Projects** page of the initial team space is displayed.

   b. Click the **Settings** icon on the menu bar, and then click **Webhooks**.

      The **Templates** tab of Webhooks is displayed.

   c. Click the **Webhooks** tab, and then click the **Disabled** webhook option to enable a webhook that is disabled.

   The event notifications that were queued when the webhook was disabled are delivered to the configured webhook or discarded by Test. This setting is based on the resume policy configuration that you set for the server webhook.

Results

You enabled the webhook on Test to push notifications of the events that failed to be delivered to the configured webhook.

Management of events

When an identified event occurs on Test , a webhook sends an HTTP POST request to the URL that you configured for the webhook. The request contains the message associated with the event, in JSON format or plain text. Each event corresponds to a set of actions that occur. For example, when a member is added to a Team space or when a Test failed with an error.

The following tables describe different events that are identified on Test.

Table 622. Main

Event	Description
All Events	When any of the events occur, a webhook sends a notification.

Table 623. Team Space Events

Event	Description
Team Space Management	When events occur at the team space management level, a webhook sends a notification.
Team Space Created	When a team space is created, a webhook sends a notification.
Team Space Deleted	When a team space is deleted, a webhook sends a notification.
Team Space Updated	When a team space is updated, a webhook sends a notification. Following are the subsets of the team space events: **Event** / **Notification sent** Team Space URL alias Updated / When the team space URL alias is updated. Team Space Renamed / When the name of the team space is changed. Team Space Becomes Permissive / When the team space is accessible. Team Space Becomes Private / When an access to the team space is not shared.

Table 623. Team Space Events (continued)

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Team Space Becomes Public</td><td>When all members are provided access to the team space.</td></tr>
<tr><td>Team Space Becomes Restrictive</td><td>When access to the team space is limited.</td></tr>
</table>
</td></tr>
<tr><td>Team Space Repositories</td><td>When an event occurs in the team space repositories, a webhook sends a notification.

Following are the subsets of the team space repository events:
<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Team Space Repositories Created</td><td>When the team space repositories are created.</td></tr>
<tr><td>Team Space Repositories Deleted</td><td>When the team space repositories are deleted.</td></tr>
</table>
</td></tr>
<tr><td>Team Space Member</td><td>When an event occurs in the team space, a webhook sends a notification.

Following are the subsets of the team space member events:
<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Team Space Member Added</td><td>When a member is added to the team space.</td></tr>
<tr><td>Team Space Member Removed</td><td>When a member is removed from the team space.</td></tr>
<tr><td>Team Space Member Role Added</td><td>When a specific role for a member is added.</td></tr>
<tr><td>Team Space Member Role Removed</td><td>When a specific role of a member is removed.</td></tr>
</table>
</td></tr>
<tr><td>Team Space Role Request</td><td>When a request is sent to join a team space, a webhook sends a notification.

Following are the subsets of the team space role request events:</td></tr>
</table>

Table 623. Team Space Events (continued)

Event	Description
	Event / **Notification sent** Team Space Role Requested — When a member sends a request to get a role in the team space. Team Space Role Request Accepted — When the request of the member for a role in the team space is accepted. Team Space Role Request Rejected — When the request of the member for a role in the team space is rejected.

Table 624. Project Events

Event	Notification sent
Project Management	When events occur at the project level, a webhook sends a notification. The Following are the subsets of project management events: **Event** / **Notification sent** Project Archived — When a project is archived. Project Created — When a project is created. Project Renamed — When a project is renamed. Project Deleted — When a project is deleted. Project Unarchived — When a project is unarchived.
Project Repositories	When an event occurs in the project repositories, a webhook sends a notification. Following are the subsets of the project repositories events: **Event** / **Notification sent** Project Repository Created — When a project repository is created. Project Repository Deleted — When a project repository is deleted.

Table 624. Project Events (continued)

Event	Notification sent
Project Member Role	When the role of a member of a project is changed, a webhook sends a notification. Following are the subsets of the project member role events: **Event** / **Notification sent** Project Member Role Added — When a project member role is added to the project. Project Member Role Removed — When a project member role is removed from the project.
Project Member Role Requests	When a request is sent to add a member role to the project, a webhook sends a notification. Following are the subsets of the project member role requests events: **Event** / **Notification sent** Project Member Role Requested — When a member sends a request to join a project. Project Member Role Request Accepted — When a request of the member to join a project is accepted. Project Member Rejected — When a request of the member to join a project is rejected.

Table 625. Execution Events

Event	Description
Test and Suite Status	Webhook sends notifications about the status of the test and test suite. Following are the subsets of the test and test suite status events: **Event** / **Notification sent** Execution Started — When you start to run a test or a test suite. Execution Stopped Manually — When the test run is manually stopped.

Table 625. Execution Events (continued)

<table>
<tr><th>Event</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Execution Completed with Verdict Pass</td><td>When the test run is completed with a pass verdict.</td></tr>
</table>
</td></tr>
<tr><td>Test and Suite Status Errors</td><td>Webhook sends notifications about the error status of the test and test suite.

Following are the subsets of test and test suite status error events:

<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Execution Completed with Verdict Fail</td><td>When the test run is completed with a fail verdict.</td></tr>
<tr><td>Execution Completed with Verdict Error</td><td>When the test run is completed with an error status.</td></tr>
<tr><td>Execution Completed without any Verdict</td><td>When the test run is completed without any results.</td></tr>
<tr><td>Execution Launch Failed</td><td>When the test launch fails.</td></tr>
<tr><td>Execution Stopped with Error</td><td>When the test run stops with an error.</td></tr>
<tr><td>Execution Stopped Automatically</td><td>When automatically the test run stops.</td></tr>
<tr><td>Execution Incomplete</td><td>When the test run is not completed.</td></tr>
</table>
</td></tr>
<tr><td>Virtual Service Status</td><td>Webhook sends notifications about the status of the virtual services.

Following are the subsets of the virtual service status events:

<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Virtual Service Started</td><td>When a virtual service is started.</td></tr>
<tr><td>Virtual Service Stopped Manually</td><td>When a virtual service is manually stopped.</td></tr>
</table>
</td></tr>
<tr><td>Virtual Service Errors</td><td>Webhook sends notifications about the errors that occur in the virtual services.

Following are the subsets of the virtual service error events:

<table>
<tr><th>Event</th><th>Notification sent</th></tr>
<tr><td>Virtual Service Launch Failed</td><td>When a virtual service launch fails.</td></tr>
<tr><td>Virtual Service Stopped with Error</td><td>When a virtual service is stopped with an error.</td></tr>
<tr><td>Virtual Service Stopped Automatically</td><td>When a virtual service is automatically stopped.</td></tr>
</table>
</td></tr>
</table>

Table 626. License Events

Event	Notification sent
License Failure	When a valid license is not obtained.

Troubleshooting

This guide describes how to analyze and resolve some of the common problems that you might encounter while you work with Test .

Known issues are documented in the Release document and the link is in the Release Notes under the Known issues section. See Defect fixes and known issues.

You can contact HCL Support if you are unable to troubleshoot the problem. Gather all the required background information and provide the details to HCL Support for investigation. For more information, see HCL Customer Support.

Troubleshooting issues

You can find information about the issues or problems that you might encounter while working with Test . Details about issues, their causes and the resolutions that you can apply to fix the issues are described.

The troubleshooting issues are presented to you in the following tables based on where or when you might encounter these issues on Test.

- Table 627: Troubleshooting issues: installation on page 1787
- Table 628: Troubleshooting issues: server administration on page 1789
- Table 629: Troubleshooting issues: configuring test runs on page 1790
- Table 630: Troubleshooting issues: test or stub runs on page 1792
- Table 632: Troubleshooting issues: Jira and Xray integrations on page 1796

Table 627. Troubleshooting issues: installation

Problem	Cause	Solution
On Ubuntu, when you are installing Test and you encounter errors in the scripts that are running.	At times, scripts might not appear to be running due to any of the following reasons: • Slow connection speeds. • Insufficient CPU, memory, or disk resources. • A firewall that was configured incorrectly is already enabled.	You can complete any of the following tasks: • To identify the issue, you can perform a diagnostic check by running the following command: `journalctl -u k3s` This command displays the log that you can use to check for the problem.

Table 627. Troubleshooting issues: installation (continued)

Problem	Cause	Solution

		• Run the following command to see which pods are running and which pods are not running:
		`kubectl get pods -A`
		Run the following command to get details about a specific pod:
		`kubectl describe pod -n <namespace> <pod name>`
		• Follow the on-screen instructions to resolve the errors.
		• Some issues can be solved by re-running the following script:
		`sudo ./k3s-init.sh`

On Ubuntu, DNS is not working as expected.		The DNS configuration that is used by the cluster can be displayed by using the following command:
		`kubectl get cm -n kube-system coredns -ojsonpath="{.data.Corefile}"`
		The **forward** setting displays the *nameservers* that are used. For example, you might see the following in the `corefile`:
		`.:53 { : forward . 8.8.8.8 9.9.9.9 : }`
		A script (`ubuntu-set-dns.sh`) is supplied for managing these values.

Table 627. Troubleshooting issues: installation (continued)

Problem	Cause	Solution

		For example, to set the DNS values for the values shown in the previous example:
		`sudo ./ubuntu-set-dns.sh --server 8.8.8.8 --server 9.9.9.9`
		Note: If you do not use sudo in the command, the script runs but the configuration might be lost if the cluster is restarted.
		To learn more about the behavior of the script, run the following command:
		`sudo ./ubuntu-set-dns.sh --help`

When running helm install the created pods keep crashing, and the logs contain: ACCESS_REFUSED when trying to connect to RabbitMQ	In some instances, the RabbitMQ password is not automatically setup correctly.	Manually apply the necessary password: `kubectl exec -n <namespace> <release-name>-rabbitmq-0 -- rabbitmqctl change_password user \ "$(kubectl get secret -n <namespace> <release-name>-rabbitmq -o jsonpath='{.data.rabbitmq-password}' | base64 --decode)"`

Table 628. Troubleshooting issues: server administration

Problem	Cause	Solution
When a user is assigned an additional role, the change in the permissions is not observed in the browser.		You must log out of the session and log in again for the changed role to take effect.
You see the following message displayed on Test:	You requested to join an project that no longer has an owner. Orphaned	Ask an administrator to take ownership of the project, and then add you as a member.

Table 628. Troubleshooting issues: server administration (continued)

Problem	Cause	Solution
`You can't request to join a project that has no owners`	projects occur when the project owners are deleted. This can occur, for example, when the person leaves the organization.	

Table 629. Troubleshooting issues: configuring test runs

Problem	Cause	Solution
When you configure a run of a schedule that matches the following conditions: • The schedule has two user groups configured to run on static agents when the schedule was created in Test Performance 10.1. • One of the user groups is disabled and the asset is committed to the remote repository. Both the static agents are displayed as available for the test run in the **Location** tab of the **Execute test asset** dialog when only one agent that is configured for the user group must be available.	The cause might be because of the following reasons: • The schedule was created in Test Performance 10.1. • The user group that is disabled is not removed or deleted from the test resources. • The agent configured on the disabled user group is already added as an agent to the server project and is available for selection.	To resolve the problem, select from either of the following methods: • By using Test Performance 10.1.1. Perform the following steps: 1. Open the schedule in Test Performance 10.1.1. 2. Save the schedule and the project. 3. Commit your test asset to the remote repository. 4. Proceed to configure a run for the schedule on Test 10.1.1. • By using Test Performance 10.1. Perform the following steps: 1. Select the disabled user group. 2. Click **Remove**. 3. Save the schedule and the project.

Table 629. Troubleshooting issues: configuring test runs (continued)

Problem	Cause	Solution
		4. Commit your test asset to the remote repository. 5. Proceed to configure a run for the schedule on Test 10.1.1.
You have added a remote repository to your project that contains the test assets or resources of the following types: • Postman • JMeter • JUnit The test assets or resources are not displayed on the **Execution** page for you to select the asset for a run.	This problem occurs if the server extension is not enabled. Although the extension was enabled at the time of installation of Test, it was disabled subsequently by the server administrator.	You must verify if the server extension is enabled and running by running the following command: kubectl get pod -n devops-system, where *devops-system* is the namespace that you created to install Test. The server extensions that are running are displayed. If the server extension that you want is not running implying that the server extension is not enabled. You must enable the server extension. Contact the server administrator to enable the server extension.
When you run AFT Suites in Test, the test run finishes with the `Completed with error` message in the **Status** column. On hover over this status, the `Pod failed reason OOMKilled` message is displayed.	The test run fails because of the higher memory consumption.	To resolve this issue, perform one of the following steps: • Increase the Execution Pod memory to 8Gi if you run the test in a container environment: `execution.template.resources.limits.memory: 8Gi` For details on the Helm parameter, see Set memory limit. • Decrease the number of tests in the AFT Suite if you want to run the test without an agent. Edit the `.xml` file of the AFT Suite in HCL DevOps Test UI (Test UI) to reduce the number

Table 629. Troubleshooting issues: configuring test runs (continued)

Problem	Cause	Solution
		of tests, and then commit the changes. • Use an agent to run the test if you cannot decrease the asset size of the AFT Suite. You can perform one of the following steps: ◦ When running the test, in the **LOCATION** tab, select an agent in the **Override** column. ◦ Edit the `.xml` file of the AFT Suite in Test UI to specify an agent in the locations section, and then commit the changes.

Table 630. Troubleshooting issues: test or stub runs

Problem	Cause	Solution
You encounter any of the following issues: • When many tests are run simultaneously on the default cluster location and you observe the following issues: ◦ Out of memory errors. ◦ Observe that the test runs are slow with a high CPU usage. ◦ The Kubernetes pods are getting evicted.	The issue is seen when any of the following events occur: • Many tests are run in parallel. • The memory that is used by the tests during the test run exceeds the allocated default memory of 1 GB. • The default memory of the container is not adequate for the test run. • Pods are evicted due to low node memory.	To resolve the problem, you can increase the resource allocation for test runs by using the arguments listed in Table Table 631: Increasing resource allocation on page 1794. You can enter arguments in the **Additional configuration options** field in the **Advanced** settings panel of the **Execute test asset** dialog when configuring a test run. **Important:** The memory settings that you configure for a

Table 630. Troubleshooting issues: test or stub runs (continued)

Problem	Cause	Solution
• When you run an AFT suite that contains multiple Web UI tests and you observe the following issues: ◦ Error stating that the browser might not be installed or the browser version is unsupported. ◦ Error stating multiple random time-outs or an internal error. • When you run VU schedules that contain multiple Web UI test or large number of virtual users or a combination of Web UI, performance, and API tests, and you observe the following issue: ◦ The test run hangs and the inadequacy of memory can be inferred as the cause in the logs.		test run is persisted for the test when ever you run it. You must use this setting judiciously. Configuring all tests for an increased memory limit might affect subsequent test runs or cause other memory issues when tests run simultaneously. In addition, in the **JVM Arguments** field under the **Advanced** settings you can set the maximum heap size for the test run time. For example, adding the JVM argument **-Xmx3g** sets the maximum heap size to *3Gi*.
You are not able to run Web UI tests on a remote agent that runs on Linux because the browser fails to launch.	Browser launch is prevented when the remote agent runs as *root* or *sudo*.	To resolve this issue, you must perform the following steps after you have installed the remote agent: 1. Stop the agent. 2. Start the agent as the logged in user by performing the following steps: a. Open a terminal on Linux. b. Enter the following commands: cd / opt/HCL/DevOpsTest/Majordomo/

Table 630. Troubleshooting issues: test or stub runs (continued)

Problem	Cause	Solution
		sudo ./MDStop.sh ./MDStart.sh

Note: You can refer to the Kubernetes documentation for information about the different units that can be used for resources in the **Additional configuration option** fields.

Table 631. Increasing resource allocation

Requirement	Configuration option name	Default value, if no value is set	An example value	Result of using the example value
Specifying the memory limit of the *init container*.	`init.resource.memory.limit`	*1024Mi*	*2048Mi*	Increases the memory limit of the *init container* from the default value to *2048Mi*.
Configuring a larger memory request for the *init container* to avoid pod eviction.	`init.resource.memory.request`	*64Mi*	*1024Mi*	Increases the initial memory request for the *init container* from the default value to *1024Mi*.
Specifying the cpu request for the *init container*.	`init.resource.cpu.request`	*50m*	*60m*	Increases the cpu request for the *init container* from the default value *60m*.
Specifying the memory limit of the container used for the test run.	`resource.memory.limit`	The larger of *3Gi* or *maximum heap size + 1Gi*	*4Gi*	Changes the memory limit of the main container from the default value to *4Gi*.

Table 631. Increasing resource allocation

(continued)

Requirement	Configuration option name	Default value, if no value is set	An example value	Result of using the example value
Note: If the memory limit you set is more than the default limit, you must run the following command to increase the allotted limit to 8 GB or any other value above the default value of 3 GB, at the time of installing the server or anytime later: `--set execution.template.resources.limits.memory=8Gi` You must be a server administrator to run this command.				
Specifying the memory request for the container used by the test run.	`resource.memory.request`	*1024Mi*	*2048Mi*	Increases the memory request for the main container from the

Table 631. Increasing resource allocation

(continued)

Requirement	Configuration option name	Default value, if no value is set	An example value	Result of using the example value
				default value to *2048Mi*.
Specifying the cpu request for the main container used by the test run.	`resource.cpu.request`	*50m*	*70m*	Increases the cpu request for the main container from the default value to *70m*.

Table 632. Troubleshooting issues: Jira and Xray integrations

Problem	Cause	Solution
You are not able to create Jira work items in Jira. You are also not able to view the created work items in Jira that you created in HCL OneTest™ Server 10.5.4 or earlier.	The cause can be due to the change in the authentication method used to connect to Jira in Test 11.0.0 or later.	Reconfigure Jira as a change management system in Test 11.0.0 or later. See Configuration of Atlassian Jira as a change management on page 1609.
You are not able to export results to the Xray project. You are also not able to view the results exported to the Xray project that you created in HCL OneTest™ Server 10.5.4 or earlier.	The cause can be due to the change in the authentication method used to connect to Jira in Test 11.0.0 or later.	Reconfigure the Xray project after you reconfigure Jira Test 11.0.0 or later. See Integration with Xray test management for Jira on page 1703.

Failed to install DevOps Test Runtime on Windows 11 25H2

When you attempt to install DevOps Test Runtime on Windows 11 version 25H2, the standard installation might fail with an error message.

The installer might display the following error message:

```
There is a problem with this Windows Installer package. A program required for this install to
 complete could not be run. Contact your support personnel or package vendor.
```

Resolution

To work around this issue, you must install the `devops-test-runtime.msi` file by using a command prompt as an administrator.

Note: You must ensure that the file name of the MSI package does not contain any spaces. For example, if you download the file multiple times, your browser might save it as `devops-test-runtime (1).msi`. Running the installation commands with spaces in the file name causes the command to fail and triggers an error dialog. To avoid this issue, you must rename the downloaded file to remove any spaces before you run the commands.

Perform the following steps:

1. Open a command prompt as an administrator.
2. Navigate to the directory where the `.msi` file is available.
3. Enter the following command in the command prompt:

   ```
   msiexec /i MSI_PATH
   ```

   The installation window with the progress bar is displayed.
4. Click **Proceed** to accept the download of the proprietary certificate if prompted.

   You have installed DevOps Test Runtime by using the command-line interface.

Related information

Recording a Web UI test on page 771

Failed to load the extension in Google Chrome while recording

When you select the Chrome browser to record a Web UI test in Test , you might get the Failed to load extension message.

Resolution

You must install the DevOps Test browser extension for Chrome that is available on the Chrome Web Store at DevOps Test Chrome extension.

You have an option to install the extension in the default profile of Chrome or use one of the other profiles available. If you use a different profile, then you must add an environment variable in your system with the following values:

- **Variable name**: DEVOPS_TEST_CHROME_PROFILE_NAME
- **Variable value**: <profile_name>

Perform the following steps:

1. Open Google Chrome.
2. Switch to the Chrome profile for which you want to add the extension by clicking the profile on the toolbar. On a Mac system, you must run the following command to create a new temporary profile by cloning the existing profile:

- mkdir -p "$HOME/Library/Application Support/devops-test-runtime/chrome-temp-profile" && cp -R "$HOME/Library/Application Support/Google/Chrome/Default" "$HOME/Library/Application Support/devops-test-runtime/chrome-temp-profile/"

And then open Chrome with the cloned profile.

3. Go to the Chrome Web Store.

   The extensions page is displayed.

4. Click **Add to Chrome**.

   The **Add "DevOps Test"?** dialog is displayed.

5. Click **Add extension**.

   A confirmation message is displayed.

6. Close all the Chrome browsers.

   Additionally, you can check and terminate any active browser processes in the background.

7. Open Test in the Edge or Firefox browser.
8. Record a test by choosing the Chrome browser from the **Browser** drop-down list.

Release

Release standardizes and automates the software release lifecycle with the DevOps Loop, moving applications smoothly from preproduction to production. It provides predictable scheduling, integrates changes from planning tools, and gives stakeholders clear visibility into plans and milestones.

Introduction to releases and pipelines

A release represents the delivery of multiple applications through one or more phases of a life cycle. A typical release includes many deployments to test environments and a small number of highly orchestrated deployments to staging and production. Release managers can define milestones that lead up to the release, as well as a plan to deliver participating applications to each phase of the life cycle.

Releases can represent all kinds of events, from major events in the life of your company to comparatively minor events, like recurring maintenance. You assign the applications, which are business-meaningful pieces of a system or software that can be independently deployed to the release. You create deployment plans, which define the tasks that must be completed when a deployment runs, for your release.

Pipelines automate releases. Pipeline stages organize inputs and jobs as your code is built, deployed, and tested. Stages accept input from either Source control management (SCM) repositories or other external applications, such as HCL DevOps Deploy (Deploy). Stages and jobs run serially and enable flow control for your work.

Home page

Use the **Home** page to view value streams, releases, and insights defined in your environment.

Access the **Home** page and user interface using the URL provided by your administrator. To access the user interface, the administrator must authenticate your credentials. After logging into Measure, the **Home** page is displayed. From the **Home** page, you can view value streams, releases, and insights.

The Home page task bar allows you to access details and work with value streams, releases, and insights. Click **Value Streams**, **Releases**, or **Insights** to view respective home pages where you can manage value streams, govern software releases, and view real-time report information for your DevOps process.

If you have administrator privileges, there are several functions available. Click **Settings** ( ) on the task bar to perform the following tasks.

- Setup the calendar
- Perform integrations and install plugins
- View license management details
- Manage user authorization
- Manage user authentication
- Configure email server
- View your profile and create User access keys (UAKs)

The **Home page** displays all recently viewed value streams and releases with each represented in respective card views. Click the required card to work with value streams and releases. Click **View all** to access the **Value Streams** and **Releases** pages.

Administering

Configuring Measure

A default Measure administrator ID is included as part of Measure. It is used to install and complete initial configuration tasks.

The Measure administrator can perform the following tasks from the administrator user interface:

- Integrations
- User authentication
- Environment mapping
- Line of business mapping

Users who are part of the Deploy system team can perform the following tasks:

- Environment mapping
- Line of business mapping

Security considerations

To ensure user data security, specify a **logout url** when authenticating users. This ensures that the user is logged out of the system and prevents another person to access their account through the cached browser data.

Configuring an email server

Configuring an email server

Before you begin

To send email, you must define an SMTP (Simple Mail Transfer Protocol) server.

This task requires that you are an Measure administrator.

About this task

You can set up a mail server for sending notifications and email.

1. On the Measure dashboard, click **Settings > Email Server**.
2. Click **Configure email server**.
3. In the **Email server** window, in the **Mail server host** field, enter the mail server URL.
   For example, `smtp.gmail.com`.
4. In the **Port** field, enter the mail server port number.
5. **Optional:** If you want to secure your email communication, check **Secure mail server connection**.
6. In the **Mail server user name** field, enter the email user account name.
7. In the **Mail server password** field, enter the user account password.
8. In the **Mail server sender name** field, enter the mail server sender name.
9. In the **Mail server address** field, enter the mail server address name.
10. Click **Save**.

Note: To delete the email settings, click **Delete Settings**.

Configuring calendar settings

Use the **Calendar Settings** page to manage the appearance of calendar events.

About this task

This task requires that you are an Measure administrator.

Use the **Calendar Settings** page to add, remove, and set the presentation style such as color for events displayed in the calendar. You can define presentation order of events for days with multiple event types. Events types include the following.

- Release events and activities
- Financial blackouts, maintenance windows, release opportunities
- Holidays

The **Preview** portion of the page displays all the defined events.

1. To access the **User authentication settings** page, click the Settings icon ( ) at the top right of the user interface. Click **Calendar**.
2. To create an event, click **Add event type** for the type of event to be added.
   There is an **Add event type** for each event type. Type the name of the event and click the check mark icon.
   a. To change the presentation style and color, place the cursor over the default style created and right-mouse click. Optionally, click the edit icon ( ).
   b. Use the left and right arrow to set the priority for the event.
      The left arrow moves the event up and the right arrow moves it down.
3. To remove an event, move the cursor to the event and click the delete icon ( ).
4. To modify an event, move the cursor to the event and click the edit icon ( ).
   There are separate edit icons for changing the event name and presentation style.

Configuring security settings

The Measure security model is organization- and team-based. The organization is the base element; teams belong to organizations. Team members manage releases, events, and deployments. Administrators manage teams and assign users to them.

The team-based security model means that teams manage releases, reports, pipelines, events, and run deployments. Only team members can create or modify team-managed objects. Users can be assigned to more than one team. Users are identified by email address. The administrator manages the security system and manages teams and users.

After Measure is integrated with HCL DevOps Deploy on page 1885, Deploy teams and applications can be imported into Measure. When a team is imported, all team members are imported. Users are identified by email address. Administrators can also create users.

Roles and permissions

A number of roles are defined in the product that can be assigned to a user or group. Roles set boundaries on activities that a user or group member can perform.

Roles are assigned to users and groups within the team context. Users or groups are assigned to a team with a specific role. The role assigned to a user is only for the team that the user is a member. A user can be a member of multiple teams and have a different role on each.

The interlocking concept of teams, roles, and permissions ensures that users have the appropriate permissions to perform their work and not affect processes outside of their assigned scope. The interaction of these three concepts provides the mechanisms to create an infrastructure that is secure and flexible.

There are two types of roles, global and team. Global roles provide permissions to access all resources of the product. Team specific roles provide permissions to access team specific resources of the product. Users with team specific roles can view only the resources associated with that team.

You can customize the roles using the following procedure.

1. On the Measure dashboard, click **Settings** , and then click **Authorization > Roles**.
2. On the **Roles** page, click **Create role**.
3. In the **Name** field, enter the name for the role.
4. In the **Description** field, enter the description for the role.
5. In the **Scope** field, select the scope for the role.

   Note: The possible values for the scope of the roles are global and team.

6. In the **Permissions** field, select the required action under each resources.
7. Click **Save**.

Note: The users with **Team manager (global)** and **Product administrator** role can create roles.

You can assign members to global roles using the following procedure.

1. Click to assign members to roles.
2. In the **Members** field, enter or select the member to assign the roles.
3. Click **Save**.

You can use to edit roles and to delete roles.

The roles with label **Custom** are customized roles. The roles with label **static** are system generated roles. **Static** roles cannot be modified or deleted.

Note: Until new users are assigned to teams, they are automatically assigned to the Default team with no assigned role. Administrators grant users elevated permissions when they assign them to roles such as Lead Developer or Release Manager.

Note: When you create a custom role and provide permissions to the deployment plan and deployment template, you must also add group view permissions.

The following tables show the available roles.

Note: For each of the roles, the underlying resources and permissions are also scoped as team or global.

Table 633. Calendar manager

Resource	Action	Description
Calendar	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete calendar settings.

Table 634. Developer (team)

Resource	Action	Description
Deployment plan	**View**	View deployment plans.
Environment	**Run**	Run team specific deployment in pipelines.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**	View team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, run, and delete team specific tasks.
Team	**View**	View teams.
Value stream	**View**	View team specific value streams and pipelines.

Table 635. Insights manager (global)

Resource	Action	Description
Dashboard	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete dashboards.
Integration	**View**	View plugin integrations.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Value stream	**View**	View value streams and pipelines.

Table 636. Insights viewer

Resource	Action	Description
Dashboard	**View**	View dashboards
Report	**View**	View reports.
Value stream	**View**	View value streams and pipelines.

Table 637. Lead developer (team)

Resource	Action	Description
Application	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete pipeline applications.
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**, **Archive**	View, create, edit, and archive deployment templates.
Environment	**Create**, **Edit**, **Delete**, **Override**, **Run**	Create, edit, delete, and override pipelines environments. Run deployment in pipeline environments.
Gate	**Edit**, **Delete**, **Approve**, **Override**	Edit, delete, approve, and override pipeline gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore releases.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete and run task.
Team	**View**	View teams.
Value stream	**View**, **Create**	View and create value streams. View pipelines.

Table 638. Lead release manager

Resource	Action	Description
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment plan templates.

Table 638. Lead release manager (continued)

Resource	Action	Description
Environment	**Run**	Run deployment in team specific pipeline environments.
Gate	**Approve**, **Override**	Approve and override team specific gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore team specific releases.
Role	**View**	View roles.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**, **Add member**, **Remove member**	View specific teams. Add and remove members to specific teams.
User	**View**	View users.
Value stream	**View**	View team specific value streams and pipelines.

Table 639. Pipeline designer (team)

Resource	Action	Description
Application	**ViewCreate**, **Edit**, **Delete**	Create, edit, and delete team specific pipeline applications.
Deployment plan	**View**, **Create**, **Edit**	View, create, and edit team specific deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment templates.
Environment	**Create**, **Edit**, **Override**, **Delete**, **Run**	Create, edit, and delete team specific pipelines environments. Run deployment in team specific pipeline environments.
Gate	**Edit**, **Delete**	Edit and delete team specific pipeline gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete team specific tasks.

Table 639. Pipeline designer (team) (continued)

Resource	Action	Description
Team	**View**	View teams.
Value stream	**View**, **Create**	View and create team specific value streams. View pipelines.

Table 640. Pipeline executor (team)

Resource	Action	Description
Environment	**Run**	Run deployment in team specific pipeline environments.
Value stream	**View**	View team specific value streams and pipelines.

Table 641. Product administrator

Resource	Action	Description
Account setting	**View**, **Edit**	View and edit account settings.
Application	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete pipeline applications.
Authentication configuration	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete authentication configurations.
Calendar	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete calendar settings.
Data retention policy	**Add policy**, **View**, **Edit**, **Delete**, **Execute**	Add, view, edit, delete, and execute the data retention policy.
Dashboard	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete dashboards.
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**, **Archive**	View, create, edit, and archive deployment templates.
Email server configuration	**View**, **Edit**	View and edit email server configurations.
Environment	**Create**, **Edit**, **Override**, **Delete**, **Run**	Create, edit, override, delete, and run pipeline environments.
External User Token	**View**, **Edit**	View and edit external tool user tokens.
Gate	**Edit**, **Delete**, **Approve**, **Override**	Edit, delete, approve, and override pipeline gates.

Table 641. Product administrator (continued)

Resource	Action	Description
Group	**View**, **Create**, **Edit**, **Delete**, **Add member**, **Remove member**, **Add role**, **Remove role**	View, create, edit, and delete groups. Add and remove members to groups. Add and remove roles to groups.
Integration	**View**, **Create**, **Edit**, **Delete**, **Upgrade**	View, create, edit, delete, and upgrade plugin integrations.
Plugin	**View**, **Create**, **Edit**, **Delete**, **Install**, **Download**	View, create, edit, delete, install, and download plugins.
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore releases.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Role	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete roles.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run tasks.
Team	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, create, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.
Teamspace	**View**, **Create**, **Edit**, **Add member**, **Remove member**	View, create, edit teamspace. Add and remove members to teamspaces.
User	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add access key**, **Remove access key**	View, create, edit, and delete users. Add and remove roles to teams. Add and remove access key to users.
Value stream	**View**, **Create**, **Edit**, **Delete**, **Design**	View, create, edit, delete, and design value streams.

Table 642. Release executor

Resource	Action	Description
Deployment plan	**View**	View team specific deployment plans.
Group	**View**	View groups.
Release	**View**	View team specific releases.
Task	**View**, **Run**	View and run team specific tasks.

Table 643. Release manager (team)

Resource	Action	Description
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore team specific deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment plan templates.
Environment	**Run**	Run deployment in team specific pipelines.
Gate	**Approve**, **Override**	Approve and override team specific gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**	View teams.
Value stream	**View**	View team specific value streams and pipelines.

Table 644. Release participant (team)

Resource	Action	Description
Deployment plan	**View**	View team specific deployment plans.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**	View team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**	View teams.

Table 645. Team manager (team)

Resource	Action	Description
Group	**View**	View groups.
Role	**View**	View roles.
Team	**View**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.

Table 645. Team manager (team) (continued)

Resource	Action	Description
User	**View**	View users.

Table 646. Team manager (global)

Resource	Action	Description
Group	**View**	View all groups.
Role	**View**	View all roles.
Team	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, create, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.
User	**View**	View all users.

Table 647. Value stream manager

Resource	Action	Description
Value stream	**View**, **Create**, **Edit**, **Design**, **Delete**	View, create, edit, design, and delete team specific value streams and pipelines.

Table 648. Value stream viewer

Resource	Action	Description
Value stream	**View**	View team specific value streams and pipelines.

Table 649. Viewer (global)

Resource	Actions	Description
Account setting	**View**	View account settings.
Authentication configuration	**View**	View authentication settings.
Calendar	**View**	View calendar settings.
Data retention policy	**View**	View the audit logs of the data retention policy.
Dashboard	**View**	View dashboards.
Deployment plan	**View**	View deployment plans.
Email server configuration	**View**	View email server configurations.
Group	**View**	View all groups.
Integration	**View**	View configured plugin integrations.

Table 649. Viewer (global) (continued)

Resource	Actions	Description
Plugin	**View**	View available plugins.
Release	**View**	View releases.
Report	**View**	View reports.
Role	**View**	View all user roles.
Team	**View**	View all teams.
Teamspace	**View**	View all teamspaces.
User	**View**	View all users.
Value stream	**View**	View deployment plans and value streams. Create user access token.

Table 650. Viewer (team)

Resource	Action	Description
Deployment plan	**View**	View deployment plans.
Group	**View**	View team specific groups.
Release	**View**	View team specific releases.
Role	**View**	View team specific user roles.
Team	**View**	View specific teams.
User	**View**	View team specific users.
Value stream	**View**	View team specific value streams and pipelines.

Role mapping limitation

In Loop, Measure uses Keycloak groups to map roles. You can refine the permissions by using the authorization features. Measure supports the following roles:

- Product administrator
- Non-admin user

These roles define the baseline access level. You can extend this baseline access by using authorization within Measure.

Role mapping matrix for Measure

Keycloak group	Loop role	Supported	Notes
User group	Product administrator	No	Users in the Keycloak user group cannot be directly assigned product administrator priviliges in Measure.
User group	Non-admin user	Yes	Standard user access is supported.
Admin group	Non-admin user	No	Admin grup users cannot be downgraded to non-admin user via Loop role mapping.
Admin group	Product administrator	Yes	Administrator access is correctly mapped from Keycloak.

Authorization behavior in Measure

You can elevate permission for users in the Keycloak non-admin user group by using the authorization feature in Measure.

When configuring these settings, note the following behavior:

- You can grant non-admin users additional access rights directly at the Measure level.
- You can manage these permissions independently of Keycloak group role mapping.
- Authorization controls what actions a user can perform within Measure, but it does not change the user's Keycloak group.
- Role mapping changes in Measure do not synchronize back to Keycloak groups.

Note:

- Keycloak mapping supports only product administrator and non-admin user roles.
- Keycloak defines baseline access, while Measure authorization extends these permissions for non-admin users.
- Role mapping does not synchronize back to Keycloak groups.

Managing users and authentications

Use **Authorization** and **Authentication** to manage users and authentication.

Users can be created, modified, or removed in Measure.

Users can be authenticated from the systems detailed in the following table:

Table 651. User authentication methods

Authentication method	UI	API
Local on page 1812	Supported	Not supported
LDAP on page 1813	Supported	Not supported
OpenID Connect (OIDC) on page 1817	Supported	Not supported
SSO on page 1817	Supported	Not supported
User access key (UAK) on page 1820	Not supported	Supported

API authentication

UAK is the recommended method through the API. An example for UAK is:

```
curl -X POST -k -H 'Content-Type: application/json' -H 'Authorization: UserAccessKey xxxxxxxxxxxxxxxxxxxx'.
```

Details for configuring UAKs are found here on page 1820.

Managing users

Use the **Users** page to create, edit, or remove users.

About this task

This task requires that you are an Measure administrator.

Users can be created, edited, or removed from **Authorization** on the **Settings** page.

To create a user, complete the following steps:

1. Click **Settings > Authorization > Users > Create user** to display the **Create user** window.
2. Complete the required fields.
3. Click **Save**.

What to do next

To modify or remove a user, click **Settings > Authorization > Users** and perform the following steps.

- To modify a user:
  - Click the overflow menu ⋮ and select **Edit** for the required user.
  - Change the required fields in the **Edit user** window.
  - Click **Save**.

- To remove a user, click the overflow menu ⋮ and select **Delete** for the required user.

  Note: Users are deleted from all teams and groups. Users' access keys are also deleted. You can use a deleted user email address to create a new user. The new user is created with default permissions. The user cannot retain previous roles and permissions after deletion.

Note: Additional user authorization topics include teams, groups, and user access keys.

Authenticating users with LDAP

Use the **User authentication settings** page to manage user authentication.

Before you begin

You must have the URL of the external LDAP server that you intend to authenticate users. This task requires that you are an Measure administrator.

About this task

An LDAP realm identifies users and groups and defines rules how to search users and groups. When unknown users attempt to log on, an external LDAP server authenticates them by using the realm parameters that you configure. To configure an LDAP authentication realm, you identify the URL of the LDAP server, and define valid searches.

To create an LDAP realm, complete the following steps:

1. From the Measure dashboard, page, click **Settings > Authentication**.
2. Click **Configure LDAP** .
3. On **LDAP** page, in the **Name** field, enter a name for the realm configuration.
   The value is an arbitrary label that does not effect the other settings.
4. In the **LDAP URL** field, enter the URL for the LDAP that you use for authentication.
   Separate multiple servers by commas.
   Example
   For example, `ldap://ldap_server.my_domain.com:389,ldap://ldap_server.my_domain2.com:389`.
5. Specify whether anonymous searches are allowed by completing one of the following options:
   Choose from:
   - If the LDAP server allows anonymous searches, select **Search Anonymously**.
   - For authenticated searches, clear the **Search Anonymously** check box, and then enter the **Bind DN** and **Bind credentials**. Velocity uses these fields to authenticate users when it connects to the LDAP server. For example, `cn=velocity,ou=applications,dc=mydomain,dc=com`.
6. **Optional:** In the **Scope when searching LDAP users** area, specify a search scope by selecting one of the following options:
   Choose from:

- **Subtree**. Select this option when user entries are direct children of the **Search base**.
- **One level**. Select this option if all user entries are direct grandchildren of the **Search base**.
- **Base**. Select this if option if user entries are two or more levels below the **Search base**.

The scope is relative to the **Search base** selected in the next step. It is a good practice to make the scope as narrow as possible.

7. In the **Search base** field, enter the user search base.
   Example
   The starting directory for the search, such as `ou=employees,dc=mydomain,dc=com`.
8. In the **Search filter** field, enter the search filter.
   Example
   The LDAP filter expression that is used when searching for user entries. The user name replaces the *{{username}}* variable in the search pattern, for example, `uid={{username}}`.
   If the value is not part of the DN pattern, enclose the value in parenthesis, for example, `(mail={{username}})`. For more information, see the help information for your LDAP server and look for information about creating user search filters.
9. **Optional:** In the **Bind property** field, enter a search expression.
   Example
   This is the name of the LDAP attribute that contains the **Bind DN** specified earlier. The default value is `dn`.
10. **Optional:** In the **Name attribute** field, enter the LDAP user name.
    This is the name of the LDAP attribute that contains the user's full name. Examples are `cn` and `displayName`.
11. In the **Email attribute** field, enter the user email address.
    This is the name of the LDAP attribute that contains the user's email address. For example, `mail`.
12. In the **Role definition** area, specify a role by completing one of the following options:

    a. Select **Role in LDAP reference their members** if you want to find group membership by searching roles, and then define the **Group search filter**.

    Note: The **Group search filter** is the LDAP filter expression that is used when you search for user entries. The user name replaces the {1} variable in the search pattern and the full user distinguished name (DN) replaces the {0} variable, for example, `(&(uniqueMember={dn})(cn=BSO*))`. The user name replaces the `{username}` variable in the search pattern and the full user distinguished name replaces the `{dn}` variable.

    a. Select **User roles are defined as an attribute on that user** if you want to find group membership by using this attribute, and then define the **Group DN Attribute** and **User Group Attribute** fields.
    The **Group DN Attribute** is the name of the LDAP attribute on group entries, whose value is the group's distinguished name. For example, `dn`. The **User Group Attribute** is the name of the LDAP attribute on user entries, whose value is the distinguished name of a group of which the user is a member. For example, `memberOf`.

13. In the **Group search base** field, enter the directory that is used for group searches.
    Example
    For example, `ou=employees,dc=mydomain,dc=com`.

14. In the **Group name attribute** field, enter the name of the entry that contains the users' group names in the directory entries that are returned by the group search.
    If this entry is not specified, no group search runs, for example, `cn`.
15. On the **Search group subtree** box, subtrees (if any) are searched. If the item is not selected, the search is limited to the **Group search base**.
16. Click **Save**.
17. Click **Import users** to verify the **LDAP** configuration settings. For the valid **LDAP** configurations, user import status is successful. For the invalid **LDAP** configurations, user import fails with the error message.

Results

The first time an unknown user attempts to log on, LDAP authorization realms are searched in an attempt to identify the user. If the user is found, a corresponding user ID is created in Measure. In addition, if the user is part of an LDAP group, that group is imported too.

When new users log on to the server and use their LDAP credentials, they are listed on the `Users` page. In most cases, do not manage user passwords or remove users from the list. If an active user is removed, they are still able to log on to the server while their LDAP credentials are valid.

Copying DevOps Deploy authentication and authorization realms into Measure

Transfer authentication and authorization realm settings from DevOps Deploy to Measure.

Before you begin
You must have both the authentication and authorization realms created in Deploy before you can copy them into Measure. This task requires that you are an Deploy and Measure administrator.

1. From the Deploy home page, click **Settings > Authentication (Users)**, and then click the required authentication realm.
2. Click the arrow to the left of **Edit** to display the authentication realm fields to be copied into the Measure fields.
3. From the Measure home page, click **Settings > User Authentication**.
4. Copy information from the DevOps Deploy authentication realm fields to the Measure fields, as described in the following table.

Table 652. Copying authentication realm fields

Deploy authentication realm field	Measure field
Name	**Name**
Description	**Description**
LDAP URL	**LDAP URL**
Search Anonymously	**Search Anonymously** Note: Match selection to Deploy field.
Search Connection DN	**Bind DN**

Deploy authentication realm field	Measure field
Search Connection Password	**Bind credentials**
User Search Base	**Search base**
User Search Filter	**Search filter** Note: Do not copy the **User Search Filter** field. In the **Search filter** field, enter `mail={{username}}`.
Name Attribute	**Name Attribute**
Email Attribute	**Email Attribute**

5. Match the **Search User Subtree** selection with the **Scope when searching LDAP users** selection.
   If **Search User Subtree** and **Subtree** are not selected, follow these guidelines for making a selection in the **Scope when searching LDAP users** field:

Options
One level indicates that only the immediate children of the entry that is specified as the search base are considered. The base entry itself is not considered, nor any descendants of the immediate children of the base entry.
Base indicates that only the entry that is specified as the search base is considered. None of its subordinates are considered.

6. From the Deploy home page, click **Settings > Authorization (Groups)**, and then click the required authorization realm.
7. Click **Edit** for the required authorization realm to display the fields to be copied into the Measure fields.
   For the following steps, you will be copying information from the Deploy authorization realm fields displayed in the **Create Authorization Realm** fields to the Measure fields.
8. Copy information from the Deploy authorization realm fields to the Velocity fields, as described in the following table.

Table 653. Copying authorization realm fields

Deploy authorization realm field	Measure field
Group Search Base	**Group Search Base**
Group Search Filter	**Group Search Filter** Note: Do not copy the **Group Search Filter** field, enter `member={{dn}}`
Group Name	**Group Name Attribute**
Search Group Subtree	**Search Group Subtree** Note: Match selection to Deploy field.
User Group Attribute	**User Group Attribute**

9. Click **Save** on the **User authentication** page in Measure.
10. Click **Import users** to verify the **LDAP** configuration settings. For the valid **LDAP** configurations, user import status is successful. For the invalid **LDAP** configurations, user import fails with the error message.

Authenticating users with OpenID Connect (OIDC)

Use the **Authentication** page to manage user authentication with **OpenID Connect**.

Before you begin

To configure the **OIDC** authentication, you must have Measure administrator permission and privileges.

About this task

OpenID Connect is an authentication method which you can use to authenticate users.

To configure an **OIDC** authentication, complete the following steps:

1. From the Measure page, click **Settings > Authentication > OIDC**.
2. Click **Configure OIDC** .
3. On **OIDC** page, in the **Name** field, enter a name for the **OIDC** configuration.
4. In the **Description** field, enter a description for the **OIDC** configuration.
5. In the **OIDC Issuer** field, enter the issuer for **OIDC** authentication.
6. In the **Authorization URL** field, enter the URL of **OIDC** authorization provider .
7. In the **Token URL** field, enter the URL for token request with **OIDC** provider.
8. In the **User info URL** field, enter the URL for user information request with the **OIDC** provider.
9. In the **Client ID** field, enter the client ID that is registered with the **OIDC** provider.
10. In the **Client Secret** field, enter the client secret that the **OIDC** provider issues upon registration.
11. Click **Save**.
    Result

    You have successfully configured **OIDC** authentication. You can log in to Measure using **OIDC** configuration.
12. On the Measure login page, click **Login with OIDC** to log in using OIDC configuration.
    The **Authorization URL** page opens to validate the user for accessing the Measure
13. On the **Authorization URL** page, enter the username in **Username** field and password in **Password** field.
    You have successfully logged in to the Measure.
14. On the **Settings > Authentication > OIDC** configuration page, click **Delete configuration** to delete the **OIDC** configuration.

Results

You have successfully configured the **OIDC** authentication. You can log in to Measure using the **OIDC** authentication.

Creating a header-based SSO authentication integration

A header-based single sign-on (SSO) authentication realm uses an external server for authentication.

1. On the Measure dashboard, click **Settings > User authentication > SSO > Configure SSO**.
2. In the required fields, specify the following parameters:

Table 654. SSO properties

Field	Description
Name	The name for the integration.
Full name header name	The user name on the SSO server that contains the list of users.
Email header name	The user email on the SSO server that contains the list of users. Note: For **Email header name**, there is a change to lower case in the Measure database.
Logout URL	The SSO server's logout URL.

Note: Measure supports header-based SSO and can integrate with other SSO mechanisms, such as Security assertion markup language (SAML). It is the responsibility of the user to modify the SSO scheme in use to transmit the headers to authenticate with Velocity Measure and then block the login page. The aforementioned method is different for every SSO product and mechanism.

3. Click **Save**.
   Result
   Setting up an SSO user does not automatically add the user on page 1812 to the Measure database and is explained in further detail below.

Results

When users log on to an SSO-configured Measure instance, their **Email header name** SSO credential is compared to the SSO server's user list. If the email address matches an existing user in the Measure database, a user will automatically be created in the database and listed as **SSO** in the **User Type** field.

Note: Once the SSO setup is completed properly by the admin user and that user logs out of Velocity, there is a loss of admin access privileges because of the inability to log back in as the admin. New users created via the SSO login only have the Viewer and Release participant permissions by default and the admin cannot change the permissions as stated previously. Possible solutions for this scenario include the following:

1. Set the Measure server to allow direct access from specific IPs that will allow the admin to login.
2. Create an SSO user with the Measure admin email address of `admin@admin.com`.
3. Perform an SSO login with a different browser while the local Measure admin navigates to **Settings > User authentication > Teams** and grants full admin privileges to the SSO user.

As a guideline, Item 3 is the recommended method for preserving admin access privileges to Measure.

To delete a current SSO configuration after it was created, click the **Delete Configuration** button on the **Settings** page for **SSO**.

Managing user groups

You can combine users into groups. When you grant permissions to the group, you simultaneously grant them to the group members. When you assign a group to a team, all group members are simultaneously assigned to the team.

About this task

Note: Groups cannot be imported from DevOps Deploy.

To create a group, complete the following steps:

1. On the Measure dashboard, click **Settings** , and then click **User authorization > Groups**.
2. On the Groups page, click **Create Group** and in the Add New Group window, enter a name for the group, and then click **Save**.
3. With the group selected, click **Add user**.
4. In the **User** list, select a user, and then click **Save**. The user is added to the **Members** list. You can add a user by typing the user's email address. The email address is used as the user name.

What to do next

If you configure an LDAP server, groups can be imported when new users log on for the first time.

Password management

Password management details how you can change the administrator password and retrieve the RabbitMQ password from Kubernetes on page 1820 and OpenShift on page 1820.

Changing the Administrator password

Use the **Users** page to change the administrator's password.

About this task

The admin user is the Measure super user. You set the initial admin password during installation. To change the admin password, complete the following steps.

Note: This task requires that you are an administrator.

1. Click **Settings > User authorization > Users** to access the **User authentication settings** page.
2. Click the **Change password** action for the admin user, and then enter your own password.

3. Enter the new password.
4. Click **Save**.

Retrieving the RabbitMQ™ password in Kubernetes™

Accessing the RabbitMQ password from Kubernetes.

1. From a command line window, run the following command to retrieve the data contained in the secret.

   ```
   kubectl get secret velocity-rabbitmq -o yaml
   ```

2. On the `password` line, copy the password.
   The password is Base64 encrypted.
   Result
   The screen capture for copying the password is shown in the following illustration.
3. Use a Base64 decoder to decode the password.

   a. For a Linux shell, run the following command to decode the password:
      Paste the password in the command line.
      Example

      ```
      echo -n '<password>' | base64 --decode
      ```

   b. The decoded password is displayed in the shell.

Retrieving the RabbitMQ™ password in OpenShift™

Accessing the RabbitMQ password from OpenShift.

About this task

To retrieve the RabbitMQ password in the OpenShift Container Platform application, use the following procedure.

1. Navigate to the OpenShift Web Console UI.
2. Navigate to the list of secrets (**Resources > Secrets**).
3. Navigate to the **velocity-rabbitmq**.
4. Click on **Reveal Secret** to display the value in the **password** field.

Managing User access keys (UAKs)

Create a key that you can use to authenticate API calls.

About this task

Until new users are assigned to teams, they are automatically assigned to the Default team in the Viewer and Participant roles. Users in these roles can view objects, such as releases and value streams, but they cannot create or edit them. Additionally, users in these roles can auto-generate user access tokens and access API endpoints with GET requests. Administrators grant users elevated permissions when they assign them to roles such as Lead Developer or Release Manager.

1. On the Settings page, click **My Profile**.
2. Click the **Create user access key** button.
3. In the **Name** field, enter the name for the user access key.
4. Click the **Create** button.
5. In the **User access key created** window, click to copy the **Access key** to the clipboard.
6. Save the key to a secure location.

   Note: It's important to save the key because it will not be accessible after you close the **Access key created** window.

7. Click **x** to close the **User access key created** window.

What to do next

You can use the key whenever you use plugins or external clients to integrate tools into Measure.

Integrations support the capability to auto-generate the required user access keys that is implemented on a plugin by plugin basis.

Note: Plugins are not updated at the same time and you can refer to the plugin documentation page for the latest updates. The update status of existing integrations only change when you edit the integration.

The auto-generated user access key process is detailed in the following:

1. On the Measure **Home** page, click **> Integrations > Plugins**.
2. Click the **Add integration** button for the required plugin integration to display the **Add Integration** modal window.
3. The **User Access Key** field is displayed if the plugin is not updated.

   Note: The **User Access Key** field is not displayed if the plugin is updated.

4. Fill in the required fields for the plugin.
5. Click the **Show Hidden Properties** check box to display the auto-generated properties for the integration.

   Note: In the Measure **User Access Key** field, you can enter a user access key if you have one, but it is not required.

6. Click the **Add** button.

7. Click **User Authorization > User Access Keys** to locate the plugin in the **Name** field with the format of **Integration Name** - AutoGen.

   Note: Any existing plugin integrations that are edited and saved automatically auto-generate user access keys in the aforementioned manner.

You can review the issued user access keys on the **User access key management** page, **Settings > User authorization > User access keys**. If you have administrator permissions, you can delete user access keys from that page.

Configuring log level settings

Control the amount of detail sent to log files.

About this task

Log files contain information about the data that your plugin integrations use or create. You can control the amount of information kept in the log files by setting the logging levels. You set the logging level separately for each integration.

The following list contains the logging levels:

- `ALL`
- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`
- `FATAL`
- `OFF`

Each level contains less information than the level that precedes it in the list. To keep the most verbose information, select `ALL`. The default logging level is `INFO`.

1. On the **Settings** page, click **Integrations**, and then select the **Edit** action for the integration that you want to change.

   Note: You can also set the logging level when you create an integration.

2. On the **Add Integration** page, select the logging level from the **Logging Level** list.
3. Click **Update**.

What to do next

To see a log file, use the **View log** action associated with the integration.

If you want to troubleshoot an integration, setting the logging level to `DEBUG` is a good place to start.

Configuration of data retention policy

Data retention policies in Measure help you manage the lifecycle of data objects by archiving or deleting resources based on scope, retention rules, and execution schedules.

Data Retention is an administrative governance feature available only for Velocity 5.2.5 or later, which allows you manage the lifecycle of data objects over time automatically. By establishing automated retention policies, Product Administrators can systematically archive or permanently delete versions and their linked resources, such as builds, inventory, and deployments that are no longer required for active operations.

You can create a data retention policy by providing the details, such as **Name**, **Description**, **Retention days**, and **Resource type**. You can also select the following policy **Mode**, which defines the action applied to the resource:

- Archive mode: This mode provides a temporary cleanup step. When resource types, such as versions, reach their archival time threshold, the system moves them out of your daily workspace. The data is removed from the active user interface so it stops cluttering your daily view. The data is still available in the backend and the administrator can still retrieve this data whenever required. The archived data continue to appear in reports such as Pipeline Snapshot, Deployment Audit, and Release Readiness reports.
- Delete mode: In this mode, the system removes the resource data permanently from the system's hard drives and databases. Because Measure maintains data integrity, deleting an object (such as a software version) will also cascade to delete all of its tightly coupled background data (such as its specific builds and deployment history). The deleted resources do not appear in reports such as Pipeline Snapshot, Deployment Audit, and Release Readiness reports.

Both archive and delete operations are soft and hard lifecycle controls, respectively, from a UI perspective, but neither operation preserves full recoverability through bulk restore. Bulk restore is not supported. However, archived resources can be restored manually.

If a resource is shared across multiple teams or applications, the system does not delete or archive the resource until all associated teams and applications are included in the selected scope. This process ensures that shared resources are not removed prematurely.

You can define the policy scope using the **Scope** field. The available scopes are as follows:

- Global: The policy is applied to all the resources across all users.
- Team: The policy is applied to only selected teams. You can select multiple teams for a single policy.
- Application: The policy is applied to only selected applications. You can select multiple applications for a single policy.

You can manage data retention policies from the **Settings > Policies > Data retention policy** page. Available actions include edit, delete, execute, and view logs. You can view the execution details by using **Audit Logs** to track policy runs and results.

You must take a database backup before creating the policies, because the reports, value stream, and insights metrics will be impacted. The linked resources of the Versions are as follows:

- Builds
- Deployments
- Inventory

Adding data retention policy

You can add a data retention policy to archive or delete resources in Measure based on defined scope and retention rules.

Before you begin

You must have the product administrator role. For more information about the roles, see Roles and permissions on page 1801.

1. On the Home page, click the **Settings** icon.
2. On the **Settings** page, select **Policies > Data retention policy**.
3. Click **Add policy**.
4. Enter the following details in the given fields:
    - Name: Add the name of the data retention policy.
    - Description: Add the description about the policy.
    - Retention days: Add the number of days for data retention.
5. Select **Mode** from the drop-down list:
    - Archive: The resource data is removed from the active user interface, so it stops cluttering your daily view.
    - Delete: The resource data is removed from the system.
6. Select **Resource** type, Version, from the drop-down list.

    Note: Currently, only the **Version** resource type is supported.

7. Select **Scope** from the drop-down list:
    - Global: This policy scope is applied to all resources across all users.
    - Team: This policy scope is applied to only selected teams.
    - Application: This policy scope is applied to only selected applications.
8. Click **Create policy**.

    The data retention policy is created as is in a disabled state, which is the default state. To enable the policy, follow the steps:

    a. Select the toggle button under the **Disable / Enable** field.
    b. Review the impacted policy records preview.
    c. Enter the policy name again to confirm enable, and then click **Confirm**.

    The policy is enabled and completed. The policy is scheduled to run every 24-hour interval.

Results

You have added the data retention policy.

What to do next

Click the icon to expand and view the following details about the data retention policy:

- Policy ID
- Resource
- Scope
- Mode
- Last execution detail
- Executed by
- Affected count
- Cutoff time

Managing data retention policy

You can edit, delete, run, and view policies from the policy page.

Before you begin

You must have the product administrator role with at least one data retention policy.

1. On the Home page, click the **Settings** icon.
2. On the **Settings** page, select **Policies > Data retention policy**.
   A list of available data retention policies is displayed.
3. Click the icon adjacent to your data retention policy to select any of the following options:

Options	Description
Edit	Update Name, Description, Retention days, Mode, Scope, or Resource, and click **Save policy**.
Delete	Click **Confirm** to delete the policy. **Note:** The deleted policy cannot be recovered.
Execute	Review the impacted policy records preview. Enter the policy name again to confirm execution and click **Confirm**. A product administrator can use this option to run the policy manually.
View logs	You can review the execution log and details of the data retention policy. Click **Download log file** to download the file.

Viewing the execution history of the data retention policy

You can view the execution history of data retention policies from the Audit Logs page.

Before you begin

You must have the Global viewer role. For more information about the roles, see Roles and permissions on page 1801.

1. On the Home page, click **Settings > Audit Logs**.
2. Filter or search for the policy that is executed as Data Retention category.
3. Review execution status and details.

Results

The system displays the execution details as follows:

- Name: Name of the policy
- Category: Category as DATA_RETENTION
- Operation: Mode of the policy, Archive or Delete
- Resource: Resource type is Version
- Scope: Scope of the policy, Global, Team, or Application
- Records affected: Number of records affected by this data retention policy
- Status: Status of the policy as completed or failed
- Triggered by: Who executed the policy
- Completed at: Timestamp of the data retention policy execution

Measure MCP Server - Tech Preview

A Model Context Protocol (MCP) server provides a structured way for GenAI clients to access tools, data, and actions through a consistent, secure interface. When you connect a GenAI client that supports MCP to the Measure MCP server, you can use natural-language requests to retrieve information from your Measure.

Disclaimer:

This release contains access to the MCP server feature in Velocity Measure as a Tech Preview and can be enabled by passing the `betaFeature=mcp` parameter while running the installer. The Tech Preview is intended for you to view the capabilities of MCP server offered by Velocity, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL® provides the information without obligation of support and "as is" without warranty of any kind.

The Velocity Measure MCP server exposes predefined queries that let you view Value Stream Management data, metrics, build and deployment details, workflow gate statuses, and information about teams, applications, and workspaces. This capability provides a simple, conversational way to obtain operational insights without requiring navigation of the Measure interface or creation of API requests.

The Measure MCP server offers the following tools:

- applicationById
- builds
- deployments
- latestWorkflowGateChecks
- metrics
- teamsForCurrentUser
- teamspacesForTenant
- workflow
- workflowsForTenant
- workflowStatsByIdQuery

To use the Measure MCP server, you must enable the MCP server feature flag when you run the installer. See Prerequisites to connect with the MCP server in Measure on page 1827.

You can currently connect to the Measure MCP server using the following GenAI clients:

- Claude Desktop. See Connecting to the DevOps Velocity MCP server in Claude Desktop on page 1828
- GitHub Copilot in Visual Studio Code. See Connecting to the Measure MCP server by using GitHub Copilot in Visual Studio Code on page 1829.

The MCP deployment is enabled, and the MCP pod is running in the default configuration. To reduce resource consumption, you must disable the MCP deployment when the MCP server is not in use. See Disabling the MCP Deployment in DevOps Velocity.

Prerequisites to connect with the MCP server in Measure

You can find information about the requirements and setup steps needed before connecting to the MCP server in Measure.

Before connecting to the MCP server, you must complete the following prerequisites:

1. Install **Keycloak 26.5.6**.
2. Download and run the script to create the necessary clients for the Velocity application and MCP server.

   Note: Download the compressed file from here: keycloak-setup script. Refer to the `README.md` file included in the bundle for instructions to run the script.

3. Install or upgrade the Velocity instance by providing the Keycloak URL, client secret, and JWKS URL for MCP.

   Here is a sample command to install or upgrade Velocity with the MCP configuration:

```
helm upgrade --install velocity -n <custom_namespace_name> \
                ./accelerate-<version>-helm.tgz \
                --set license=accept \
                --set url.domain= <domain_name or I.P address>\
                --set access.key=<access_key>\
                --set mongo.url=<mongodb url>\
                --set secret.tls=<tls_secrete_name>\
```

```
                --set rabbitmq.nodePort=31672 \
                --set loglevel=ALL \
                --set externalKeycloak.url=https://<keycloak_hostname>/realms/devops-velocity \
                --set
 externalKeycloak.jwksUrl=https://
<keycloak_hostname>/realms/devops-velocity/protocol/openid-connect/certs \
                --set mcp.secretValue="<mcp-velocity-client-secret>"
```

Note: `--set jwksUrl=externalKeycloak.jwksUrl=https://<keycloak_hostname>/realms/devops-velocity/protocol/openid-connect/certs` is required only for Velocity 5.2.4 and earlier versions.

Before running the Helm command, you must replace the following placeholder values with your actual configuration:

- **<keycloak_hostname>**: Replace with your Keycloak server hostname or IP address.
- **<mcp-velocity-client-secret>**: Replace with the secret value configured for the **mcp:velocity** client in Keycloak.

These parameters ensure that Velocity can authenticate users via Keycloak and connect to the MCP server securely.

For more information about installation and upgrading, refer to the following related topics:

- Installing on Kubernetes
- Installing on OpenShift
- Installing on Google Kubernetes Engine
- Upgrading Kubernetes-based installations
- Upgrading an OpenShift installation using Helm charts

Connecting to the DevOps Velocity MCP server in Claude Desktop

You can connect to the Measure Model Context Protocol (MCP) server by using the Claude Desktop application to interact with Velocity and retrieve the necessary data and insights.

Before you begin

You must have completed the following tasks:

- Installed Velocity and configured Key cloak. See Prerequisites to connect with the MCP server in Measure on page 1827.
- Configured OIDC authentication in Velocity. See Authenticating users with OpenID Connect (OIDC) on page 1817
- Obtained the MCP Server URL. For example: `https://<velocity_hostname>/mcp`
- Ensured that Claude Desktop is installed.

1. Open the **Claude Desktop** application.
2. Navigate to **Settings > Connectors**.
3. Click **Add custom connector**.
   The **Add custom connector** window is displayed.
4. Enter the name of the connector. For example: Velocity MCP server
5. Enter the Velocity MCP server URL in the **Remote MCP server URL** field.

6. Click **Add**.
   The MCP server is now added and listed on the **Connectors** page.
7. Click the **Connect** button next to the newly added Velocity MCP server entry.

   Claude Desktop redirects you to the **DevOps Loop** login page.
8. Sign in using the Velocity OIDC user credentials.

   A consent screen is displayed requesting access to MCP tools, Velocity APIs that are used internally for tool execution, and basic profile and role information.
9. Click **Yes** to approve and complete the connection.

   After successful authentication, Claude Desktop displays the list of available MCP tools provided by the Velocity MCP Server.

   Note: The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session expires and you are required to re-authenticate.

Results

You have successfully connected to the Velocity MCP server by using Claude Desktop.

What to do next

You can run prompts related to Velocity to confirm a successful connection.

Connecting to the Measure MCP server by using GitHub Copilot in Visual Studio Code

You can connect to the Measure Model Context Protocol (MCP) server by using GitHub Copilot in Visual Studio (VS) Code to interact with Measure and retrieve the necessary data and insights.

Before you begin

You must have completed the following tasks:

- Installed Measure and configured Key cloak. See Prerequisites to connect with the MCP server in Measure on page 1827.
- Configured OIDC authentication in Measure. See Authenticating users with OpenID Connect (OIDC) on page 1817
- Obtained the MCP Server URL. For example: `https://<velocity_hostname>/mcp`
- Ensured that you have installed VS Code and the GitHub Copilot extension.

1. **Optional:** Open the terminal and enter the following command to allow the self-signed certificates if you are using a self-signed certificate during the Measure installation.

   ```
   export NODE_TLS_REJECT_UNAUTHORIZED=0
   ```
2. Enter the following command to open VS Code.

   ```
   code .
   ```

3. Open the **Command Palette** by pressing the keys Ctrl+Shift+P or Cmd+Shift+P.
4. Search and select **MCP: Add Server**.
5. Choose **HTTP** as the server type, when prompted for the transport type.
   The Velocity MCP Server uses HTTP-based transport.
6. Enter the **MCP Server URL** in the **URL** field.
   VS Code automatically generates the required `mcp.json` configuration.
7. Enter a name for the server. For example: Velocity MCP Server.
8. Select **Workspace**, when VS Code prompts you to choose where to save this configuration.
   - **Global** – used for all VS Code workspaces
   - **Workspace** – used only in the current folder/project

     Note: When **Workspace** is selected, a `.vscode` folder is automatically created, and an `mcp.json` file is generated within it.

     A sample `mcp.json` file is as shown below:

     ```
     {
      "servers": {
       "velocity-mcp-server": {
        "url": "<velocity_mcp_server_url>",
        "type": "http"
       }
      },
      "inputs": []
     }
     ```

Property	Description
<velocity_mcp_server_url>	Specifies the full Velocity MCP server URL. For example: `https://<velocity_hostname>/mcp`

   Result

   The `mcp.json` file is opened.
9. Click **Start** for the added MCP server.
   Result
   A prompt appears to authenticate via OAuth.
10. Click **Allow**.
    Result
    The following prompt is displayed:

    ```
    Do you want code to open the external website?
    ```

11. Click **Open**.

Result

The OIDC login page is displayed.

12. Sign in using the Measure OIDC user credentials.

    Result

    A consent screen is displayed requesting access to MCP tools, Measure APIs (used internally for tool execution), and basic profile and role information.

13. Click **Yes**.

    You are automatically redirected to VS Code or you can also close the browser and switch to VS Code.

14. Verify the connection.

    The Measure MCP Server must display **Running** as the status.

> **Note:** The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session expires and you are required to re-authenticate.

Results

You have successfully connected to the Measure MCP server by using GitHub Copilot in VS Code.

What to do next

You can run prompts related to Measure to confirm a successful connection.

Releasing Software

A release is a repeatable plan that is used to drive deployments.

Releases are the organizing focus of software deliveries. Releases can represent any software-development-related event, from major milestones in the life of the company to comparatively minor events, like recurring maintenance. You can schedule a release at any point in your development-operations lifecycle. You can automate release schedules or manually run deployments.

A release contains deployment plans and events that are related in some business-meaningful way. For example, the deployment plans might represent the phases in a software development lifecycle, such as development, QA, and production. An event might represent a blackout that affects the release.

Every value stream has a corresponding pipeline generated for it, and every pipeline has deployment plan templates. The tight relationship between pipelines and deployments makes it easy to use pipelines to drive deployments. You can start by creating a release and adding deployment plans and events to it. Or you can start with value streams and pipelines and then join them to a release. You can run deployments directly from a pipeline or from a deployment plan.

You can coordinate several releases into a single event, which ensures that all release stakeholders can participate. If you add events to a release, you can use the calendar to organize releases and deployment plans.

Managing a release

Use the release details page to view details about a specific release.

This page contains the calendar and list of tasks. When viewing tasks you can select to view only your tasks or all tasks.

You can also use this page to access and work with release templates.

Use this page to view and work with archived releases removed from the calendar and list of tasks.

When you access a release from the **Releases** page, you can create a deployment plan and display overview details about the release, such as the deployment plans associated with the release with related details. From the **Overview** section of the **Releases** page, you can view the following release details for specific releases:

- **Plans** displays the plans associated with the release and task status, team, and date that the task is scheduled. You can access additional details for specific plans.
- **Task progress** displays the associated tasks that are not completed and contains information such as type of task, task owner, start and end time, duration, and task dependencies.
- **Approvals** list the approvals with the status of the approval and approver.
- **Change requests** displays a list of ServiceNow™ change requests.

The **Properties** section displays properties defined in the plan and property value.

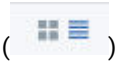
Managing releases

Release management is the process of planning, scheduling, and controlling software deliveries.

About this task

The **Releases** page displays information about releases. This page provides access to the calendar, tasks, events, release activities, and related details.

The page contains the calendar or list of tasks. Click **Calendar** to view the calendar. To hide the calender and task bar, click . To view tasks, click **Tasks**. When viewing tasks, you can select to view only your tasks or all tasks. To view events, click **Events**. To create an event, click .

Release activities are displayed in a tile or list () format. Using the search bar on the top of the page, you can limit the amount of information displayed by typing a filter to narrow the search results. Additionally, click the filter ( ) to limit the list of displayed release activities to: **Complete**, **In progress**, or **Scheduled**. Click an item in the **Release activities** list to view specific release details.

Click the tag ( ) to toggle on or off the display of tags associated with the release activity.

To create a new release, click **Create release**.

To import the releases and events, click **Import Release**.

You can also use this page to access and work with release templates by clicking **Templates** on the page.

Click **Archived** to view and work with releases removed from the **Releases** page and the calendar.

You can view release details, such as deployment plans associated with the release, by clicking **Overview**. This page will display headings that include: **Plans**, **Task progress**, **Approvals**, and **Change requests**.

- Plans associated with the release an be viewed by clicking **Plans**. The number above the **Plans** heading is the total number deployment plans related to this release.
- Tasks details can be viewed by clicking **Task progress**. The number above the **Task progress** heading displays the number of associated tasks not completed and total number of tasks. Click the task for additional task information.
- Click **Approvals** to display approvals details. The number above the **Approvals** heading is the total number of approvals. Click the approval name for details about a specific approval.
- Click **Change requests** to display a list of ServiceNow™ change requests. The number above the Change requests heading is the total number of ServiceNow™ change requests.

Click **Properties** to display a properties defined in the plan and property value. You can add or change the property value by double-clicking the **VALUE** property field. Click the check mark to save the value.

To create a new deployment plan, click **Create deployment plan**.

Creating releases

1. On the **Releases** page, click **Create release**
2. In the **Create release** window, type a name for the release in the **Name** field.
   This field can be referenced with the `${sys:release.name}` system property. A name is required.
3. In the **Tags** list, select one or more events or tags. You can also create a tag by typing the name of the new tag in the field.
   If you want the event to appear on the calendar, select an event that is created on the **Configure Calendar** page. Tags that you create with the **Create release** window are not displayed on the calendar, which can prevent calendar clutter.
4. In the **Description** field, type a description of the release. This field can be referenced with the `${sys:release.description}` system property.
5. In the **Deployment plan templates** field, select one or more templates to add to the release.
   Each template that you select, adds a deployment plan to the release with the name **Copy of *template_name***. You can convert any deployment plan into a template. If the selected templates have user-defined properties, you are prompted to provide values for them when you save the release.
6. In the **Start Time** field, select a starting date and time. This field can be referenced with the `${sys:release.start}` system property. Starting times are required for all releases.
7. In the **End Time** field, select the ending date and time. This field can be referenced with the `${sys:release.end}` system property.
8. In the **Team** list, select a team to manage the release. You can add deployment plans from other teams to the release event, not just the team that manages the release. All the teams that you belong to are available. A team is required.
9. If the selected templates have properties, click **Next**, and then provide values for the properties. Otherwise, click **Save** and view details to save the release and open the **Releases** details page. If you added templates to the release, a

deployment plan for each template is created and displayed. If you didn't select any templates, an empty deployment plan is added to the release with the name **Plan: *release_name***.

To display the release in the **Activities** list, ensure that the date range includes dates within the release's start or end times.

What to do next

After the release is created, you can add deployment plans, events, and tags to it.

Archiving releases

You can archive a release to remove it from the list of releases.

About this task

Although you cannot delete a release, you can store it in the archives. Archived releases are not displayed on the **Releases** page.

To archive a release, click the overflow ( ) and then select **Archive**.

Repeat this action for each release to be archived.

You can view archived releases and restore them by clicking **Archived** on the **Releases** page.

Events and tags

An event represents a release-related activity that you manage with the calendar. A tag is a label that you can add to a release or an event. You can use tags to filter the `Activities` list on the **Releases** page. Tags do not appear on the calendar.

Events are organized into general categories that include releases, holidays, and maintenance windows. You can add events to releases or use them independent of releases. When an event is assigned, it is represented on the calendar by an icon appropriate for the event category. You can use the icons to filter the **Releases** page and select releases for any affected date or range of dates.

Tags are assigned to releases and can be used to filter the **Releases** page. Assigned tags appear in Releases page but they do not appear on the calendar. You can use tags to avoid calendar clutter.

Restoring archived releases, plans, and templates

You can restore archived releases, deployment plans, and templates.

About this task

A restored release retains all of its original deployment plans.

1. On the **Releases** page, click **Archived**.
   The **Archive** page lists all archived items. The **TYPE** column displays whether the item is a release, plan, or template.

2. Locate the item to be archived.
   Using the search bar, you can limit the amount of information displayed by typing a filter to narrow the archive search results. Filters can contain keywords, teams, tags, or name. You can also click on some of the table headers to sort the table in ascending or descending order based on the heading selected. For example, clicking **ARCHIVED DATE**, list the items from earliest archive date to current. Click the tag ( ) to toggle on or off the displaying of tags associated with the release activity.
3. If an item needs to be restored, click the overflow ( ) and then select **Restore**.
   Repeat this step for each release to be restored.
4. Click **Releases** to return to the **Releases** page.

Results

- Restored releases are displayed in the list of releases.
- Restored releases retains all of its original deployment plans.
- Restored deployment plans have a status of **draft**.

Deployment plans

A deployment plan is a container for tasks. Tasks define the activities that your team runs to complete a software deployment.

You create a deployment plan by assigning a team to manage it and then adding tasks to the plan. Team members add tasks to the plan and run the tasks during deployments. The managing team also determines which HCL DevOps Deploy (Deploy) applications are available. After an integration is configured, all applications that are managed by the team in Deploy are available.

When you use the Pipeline feature, releases and deployment plans are automatically generated as you add stages and applications to the pipeline. You can start deployments directly from the pipeline, or open the deployment plan and start the deployment by running an eligible task.

The tasks define the activities that the team performs to complete a deployment. You can create new tasks, copy tasks from other deployment plans or import tasks from CSV files. When you are satisfied with the plan, you can schedule a deployment. You can also run a deployment at any time by starting one of the plan's tasks.

Tasks are displayed on individual rows in deployment plans. Each row contains information that identifies the task and provides its status. Each row also contains action icons that are used during deployments, such as **Start** task or **Skip**.

You can run the tasks sequentially or parallely beginning with the first task, or topmost task, in the plan. After the first task finishes, the second task is eligible to run, and so forth. The order in which tasks are eligible to run is called the execution order.

The execution order can be modified by combining tasks into groups. When you create a group on page 1849, you set the group's execution pattern, which can be sequential or parallel. If a group has a sequential execution pattern, its tasks run sequentially starting with the first task. If a group has a parallel execution pattern, tasks in the group can be started in any

order and they can run simultaneously. If a plan consists entirely of parallel tasks, you can start any task regardless of its position in the list.

Task eligibility can also be effected by task dependencies. If a task has any prerequisites on page 1848, it cannot start, even if it is otherwise eligible, until all prerequisite tasks are resolved.

Some tasks start automatically as soon as they are eligible to run, while other tasks are started manually. Whether a task starts automatically or not depends on its type. Manual tasks, as the name implies, are started manually. Deploy and Delayed-type tasks automatically start as soon as they are eligible.

Eligible tasks display a **Start Task** button. To start a deployment, you start one of a plan's eligible tasks. Scheduled deployments start automatically at the scheduled time if an eligible task is of a type that can start automatically, such as Deploy.

Managing deployment plans

You can change existing deployment plans and view current status from the **Deployment Plan** details page.

About this task

The **Deployment Plan** details page displays a list of tasks, properties, versions, and the change history associated with the plan.

- Click **Tasks** to work with the tasks in the plan. You can add tasks, schedule tasks, skip a task, and reopen a completed task.
- Click the **Properties** to view properties and their value.
- Click the **Versions** to view versions.
- Click the **Change history** to view a list of the changes, revision number, user who made the revision, and date. You can also restore a plan to a specific version by clicking restore ( ).

From this page you can modify and archive the plan, change the name of the plan, make a copy of the plan, and create a template from the plan. Access these features from the overflow menu ( ) at the top right corner of the page.

- Click **Import tasks** to incorporate tasks into the deployment plan from various file sources including spreadsheets, project management tools, or IBM® DevOps Release.
- Click **Estimate task times** for approximating task durations.
- Click **Edit** to modify the deployment plan. With this option, you can add events and tags to the release event.
- Click **Copy** to copy a deployment plan. A copy of the selected plan is added to the release.
- Click **Copy as template** to create a template based on this plan.
- Click **Archive plan** to remove the deployment plan from the release. Removed plans are not permanently deleted and can be added to another release.

Creating deployment plans

You can start with a blank plan that has no defined tasks, copy an existing plan, or start with a plan template. You can also auto-create them by assigning applications to pipelines. When you copy a plan or use a template, many tasks are already defined.

About this task

Complete the following steps to create a deployment plan:

1. On the **Releases** details page, click **Create Deployment Plan**.
2. On the Create Deployment Plan window, in the **Deployment plan template** list, specify the template that you want to base the plan on. The default value is **None**.
3. In the **Plan name** field, enter the plan name.
4. In the **Description** field, enter the description.
5. In the **Team** list, select a team to manage the plan. All teams that you are a member of are available. Team members can modify the plan and manage tasks. The HCL DevOps Deploy applications that are managed by the team in Deploy are available to assign to Deploy type tasks.
6. To schedule a deployment for the plan, click **Start time**, and then select a date and time for the deployment. Scheduled deployments automatically start at the scheduled time if the plan contains eligible auto tasks. You can schedule a deployment for any future time.
7. In the **Tags** list, select tags that you want to add to the plan.
8. Click **Save**. If you created the plan on a **Releases** details page, the plan is part of the selected release. After you save the plan, click **Edit** to modify plan details.

Note: You can create one deployment plan as a primary linked to many secondary deployment plans. Deployment is executed only from primary deployment plan.

Importing tasks into deployment plans

You can import tasks that are defined in comma-separated values (CSV) files into a deployment plan.

About this task

You can import tasks from CSV files that are exported from IBM® DevOps Release or another application capable of creating CSV files. When you import a deployment plan from Release, segments are converted into groups with the same execution pattern as the original segments. Segment tasks are bracketed by note-type Start Segment and End Segment tasks. Task dependencies are preserved during import. Segment dependencies are represented by dependencies to End Segment tasks.

To learn about deployment plans exported from Release, download the `SamplePlan.csv` file from the Import Tasks dialog.

Note: When you import tasks, tasks that are already defined in the deployment plan are overwritten and replaced by the tasks in the CSV file.

The following table contains the fields that define a task in an Release deployment plan. The first row in the CSV file contains the column headers, which correspond to the table's Field column. The rows after the first row contain task data, one task per row. The order of the columns does not matter.

Note: If you are creating tasks in a CSV file, you must include the required fields that are identified by the asterisk character (*) in the following table.

Table 655. Release deployment plan fields

Field	Description
segmentName*	The name segment. Required field.
name*	The name of the deployment task. Required field.
type*	Task type. Possible values are note, waitfortimetask, ucdtask, or manual. During import, if the field contains an unrecognized value, the task is assigned the manual type.
description	Description of the deployment plan.
property:processId	The process ID in HCL DevOps Deploy (Deploy) that corresponds to the task.
property:task-environments	Application environments available to the task.
property:mailRecipients	Email addresses of users who receive email messages when notifications are sent.
taskTagNames	The names of the tags that are associated with the task.
taskPrequisiteIds	A comma-separated list of IDs for tasks that must complete before this task can start.
segmentPrerequisiteIds	A comma-separated list of IDs for segments with which the current segment has dependencies.
taskPrequisiteNames	A comma-separated list of names for tasks that must complete before this task can start.
segmentPrerequisiteNames	A comma-separated list of names for segments with which the current segment has dependencies.
assignedUserEmail	Email address of the user who is assigned to the current task.
executorGroup	User group assigned to the current task.

Table 655. Release deployment plan fields (continued)

Field	Description
segmentPattern	Execution pattern of the selected segment.

Complete the following steps to import tasks into a deployment plan:

1. On the **Deployment Plan** details page, click **Import Tasks**.
2. In the Import Tasks dialog, click **Choose File**, and then select the CSV file in addition to confirming the choice in the dialog box. You can also drag a file onto the field.
3. Click **Import**. The tasks are added to the deployment plan. If the import process is unsuccessful, click **View Report** to display the error log.

Copying deployment plans

You can duplicate deployment plans by copying them.

About this task

The status of a copied or promoted plan is `draft` even if the original plan has a status of `done` or `in progress`.

1. **Optional:** To copy a plan, click the overflow icon ( ) and select **Copy plan** from the drop-down on the **Releases** details page. When you copy a plan, a new plan with the name "Copy of `plan_name`" is inserted into the **Releases** details page. The copied plan contains all the tasks that are defined in the original plan. If HCL DevOps Deploy (Deploy) tasks are in the plan, the applications, versions, and environments that are selected in the original are also selected in the copied plan. Applications, versions, and environments are displayed on the **Versions** tab on the Deployment Plan Detail page.
2. **Optional:** To create a template from a plan, click the overflow icon ( ) and select **Copy as template** from the drop-down on the **Releases** details page. When you copy a plan as a template, a new plan with the name "Copy of `plan_name`" is inserted into the **Templates** page. The copied plan contains all the tasks that are defined in the original plan. If Deploy tasks are in the plan, the applications, versions, and environments that are selected in the original are also selected in the template.

Managing deployment plan templates

You can convert deployment plans to create deployment plan templates.

About this task

To create a template, click the overflow icon ( ) and select **Copy as template** from the drop-down on the **Releases** details page. When you create a template, a new plan with the name "Copy of plan_name" is inserted into the **Templates** page. The status of a template is `template` even if the original plan has a status of `done` or `in progress`. The template contains all the tasks that are defined in the original plan. If HCL DevOps Deploy (Deploy) tasks are in the plan, the applications selected in the original are also selected in the template.

You cannot run deployments with templates. To use a template for a deployment, first copy or promote it and then use the copied or promoted plan.

Archiving deployment plans

You can remove deployment plans from releases by archiving them.

About this task

Deployment plans cannot be permanently deleted, but can be archived. When you archive a deployment plan, it has a status of **Archived** and is not displayed in the list of plans on a **Releases** details page.

You can view archived deployment plans and restore them by clicking **Archived** on the **Releases** page.

1. Click **Plan** for a release to view a list of associated deployment plans.
2. On the line of the plan to be archived, click the overflow menu icon ( ) and select **Archive this plan**.
   Repeat this step for each plan to be archived.

Running deployments

Complete deployments by resolving the tasks in a deployment plan.

A deployment starts when one of the plan's eligible tasks starts. A scheduled deployment starts automatically at the scheduled time if one of its eligible tasks is an auto task, such as HCL DevOps Deploy (Deploy) tasks. Otherwise, begin a deployment by starting one of the plan's eligible tasks. If the plan contains several eligible tasks, you can start any of them. You can manually start a deployment at any time. You can manually start a scheduled deployment before its scheduled time.

The deployment plan's execution pattern determines when tasks are eligible to start. If the plan has the sequential execution pattern, which is the default pattern, tasks become eligible in the order in which they are listed in the plan, starting with the first task. After the first task resolves, the second task becomes eligible. Start eligible manual tasks by clicking the task's **Start** button. Auto tasks, such as Deploy tasks, start automatically as soon as they become eligible.

The tasks in a group with a parallel execution pattern become eligible simultaneously. Tasks in parallel groups can be started in any order. Nested groups and tasks with ad hoc dependencies can affect the execution pattern.

You can modify a deployment plan after you start a deployment. You can add, delete, and modify tasks.

After all tasks are resolved, the deployment is complete. A task is resolved if it has a status of `Complete`, `Failed`, or `Skipped`. If you reopen a task or add a task after the deployment is complete, the deployment's status changes to `In Progress`.

Starting deployments manually

Starting deployments manually

About this task

You can start a deployment manually at any time, including deployments that are scheduled to start later.

Tasks in a deployment plan might be inapplicable for deployment. HCL DevOps Deploy (Deploy) tasks are inapplicable if no version or environment is specified for the tasks. When you create Deploy tasks, you can delay selecting the environment and version by using the **Use Version** option. Before the deployment starts, ensure that all Deploy tasks are applicable for the upcoming deployment.

You can deliberately exclude tasks from a deployment. You can make tagged tasks inapplicable by excluding their tags from the deployment. Tasks with excluded tags are inapplicable for deployment.

Inapplicable tasks have the status of `Not Applicable`. Tasks with the `Not Applicable` status cannot be started. If an inapplicable task is a prerequisite for an active task, the dependency is ignored.

To start a deployment, complete the following steps:

1. On the **Releases** details page, click the deployment plan that you want to use for the deployment. The **Deployment Plan** details page is displayed.
2. **Optional:** To make tagged tasks inapplicable for the current deployment, click **Versions**, and then in the **Tags** area, clear the tags. If a task has more than one tag, clear all of them.
3. To select the version or environment for any inapplicable Deploy tasks, click **Versions**, and then select the environment and version. You can also change your choices for tasks that have their environments and versions already selected.
4. **Optional:** Click **Hide Not Applicable Tasks** to remove inapplicable tasks from the displayed task list. Hidden tasks are not removed from the deployment plan.
5. Click the **Start** action for one of the plan's eligible tasks. If more than one task is eligible, you can start any of them. Only eligible tasks display the start button. A started task has the status of `In Progress` until it is resolved.
   Result
   After a deployment starts, the plan's status changes to `In Progress`. The status remains `In Progress` until all tasks are resolved. When all tasks are resolved, the plan's status changes to `Done`.

Resolving tasks

Complete a deployment by resolving the tasks in the deployment plan.

About this task

A started task is resolved when its status changes to `Complete`, `Failed`, or `Skipped`.

You resolve manual tasks by using the appropriate status action. Auto tasks, such as HCL DevOps Deploy (Deploy) tasks or Jenkins®, change status automatically as conditions change. However, you can manually resolve auto tasks. For example, you might manually fail an Deploy task that is taking too long to complete.

To resolve a started task, apply one of the following statuses:

- Click **Complete** to finish the task. `Complete` means that the task finished with the expected results.
- Click Skip to skip the task for the current deployment.

- Click **Fail** to end the task. `Fail` means that the task did not finish or finished with unexpected results.
- Click **Task Reopen** to open a resolved a task. If all tasks are completed, reopening a task reopens the deployment.

Importing tasks into deployment plans

You can import tasks that are defined in comma-separated values (CSV) files into a deployment plan.

About this task

You can import tasks from CSV files that are exported from IBM® DevOps Release or another application capable of creating CSV files. When you import a deployment plan from Release, segments are converted into groups with the same execution pattern as the original segments. Segment tasks are bracketed by note-type Start Segment and End Segment tasks. Task dependencies are preserved during import. Segment dependencies are represented by dependencies to End Segment tasks.

To learn about deployment plans exported from Release, download the `SamplePlan.csv` file from the Import Tasks dialog.

Note: When you import tasks, tasks that are already defined in the deployment plan are overwritten and replaced by the tasks in the CSV file.

The following table contains the fields that define a task in an Release deployment plan. The first row in the CSV file contains the column headers, which correspond to the table's Field column. The rows after the first row contain task data, one task per row. The order of the columns does not matter.

Note: If you are creating tasks in a CSV file, you must include the required fields that are identified by the asterisk character (*) in the following table.

Table 656. Release deployment plan fields

Field	Description
segmentName*	The name segment. Required field.
name*	The name of the deployment task. Required field.
type*	Task type. Possible values are note, waitfortimetask, ucdtask, or manual. During import, if the field contains an unrecognized value, the task is assigned the manual type.
description	Description of the deployment plan.
property:processId	The process ID in HCL DevOps Deploy (Deploy) that corresponds to the task.
property:task-environments	Application environments available to the task.
property:mailRecipients	Email addresses of users who receive email messages when notifications are sent.
taskTagNames	The names of the tags that are associated with the task.

Table 656. Release deployment plan fields (continued)

Field	Description
taskPrequisiteIds	A comma-separated list of IDs for tasks that must complete before this task can start.
segmentPrerequisiteIds	A comma-separated list of IDs for segments with which the current segment has dependencies.
taskPrequisiteNames	A comma-separated list of names for tasks that must complete before this task can start.
segmentPrerequisiteNames	A comma-separated list of names for segments with which the current segment has dependencies.
assignedUserEmail	Email address of the user who is assigned to the current task.
executorGroup	User group assigned to the current task.
segmentPattern	Execution pattern of the selected segment.

Complete the following steps to import tasks into a deployment plan:

1. On the **Deployment Plan** details page, click **Import Tasks**.
2. In the Import Tasks dialog, click **Choose File**, and then select the CSV file in addition to confirming the choice in the dialog box. You can also drag a file onto the field.
3. Click **Import**. The tasks are added to the deployment plan. If the import process is unsuccessful, click **View Report** to display the error log.

Running scheduled deployments

A scheduled deployment starts automatically at its scheduled time if any of its auto tasks are eligible to start.

About this task

If a deployment plan does not contain any eligible auto tasks, start the deployment by manually starting one of the eligible tasks. If the deployment plan belongs to a pipeline-generated release, you can start the deployment directly from the pipeline.

You can manually start a scheduled deployment at any time even if the plan has eligible auto tasks.

Tasks

A task represents a business-meaningful activity that is associated with a software deployment. Each task is defined on a separate row in a deployment plan. During a deployment, you run tasks defined in the deployment plan to complete the deployment.

Most tasks have a starting and ending point, and a measurable duration. A task can have other attributes depending on its type. A task can be of one of following types:

Ansible

Automation tool used for configuration management, application deployment, intraservice orchestration, and provisioning tasks.

Delayed

Represents a critical event that happens at a specific time.

Email

Sends an email message.

Header/Note

an organizational element. For example, you might use a header task to identify a task group.

Jenkins®

Runs a Jenkins® job or pipeline.

Manual

Represents any activity that is associated with a software deployment, such as taking a server offline or updating a database. Users assigned manual tasks can be sent email notifications when their tasks start or fail.

Run a Shell Script

Shell script tasks are used to execute a sequence of commands. The shell is the command line interface (CLI) for the operating system (OS) used to interpret the sequence of commands communicating with the software interface.

Run another plan

Runs deployments for other deployment plans participating in the same release event. This is an experimental task type.

ServiceNow™

Creates and manages ServiceNow tickets.

HCL DevOps Deploy (Deploy) (requires Deploy integration)

Use this task to run an Deploy application process.

Deploy Status (requires Deploy integration)

Use this task to add a status to all component versions contained within an Deploy snapshot.

Wait for approval

Pauses a deployment until the designated users approve the task.

You can add tasks to deployment plans by one of the following methods:

- Create a new task
- Import from CSV files that are created by Deploy or another application
- Copy tasks from other deployment plans

Additionally, tasks are auto-created when you add an application to a pipeline on page 3127.

Managing tasks

A list of associated tasks can be viewed from a **Deployment Plan** details page. The list of tasks include details about the task. You use action icons to change the task's status during a deployment, such as start or skip a task.

About this task

Each task in a deployment plan is contained in a separate row. Click ( ) to show or hide associated tasks from the list.

Click ( ) show or hide tags.

The information that is displayed for each task is described in the following table.

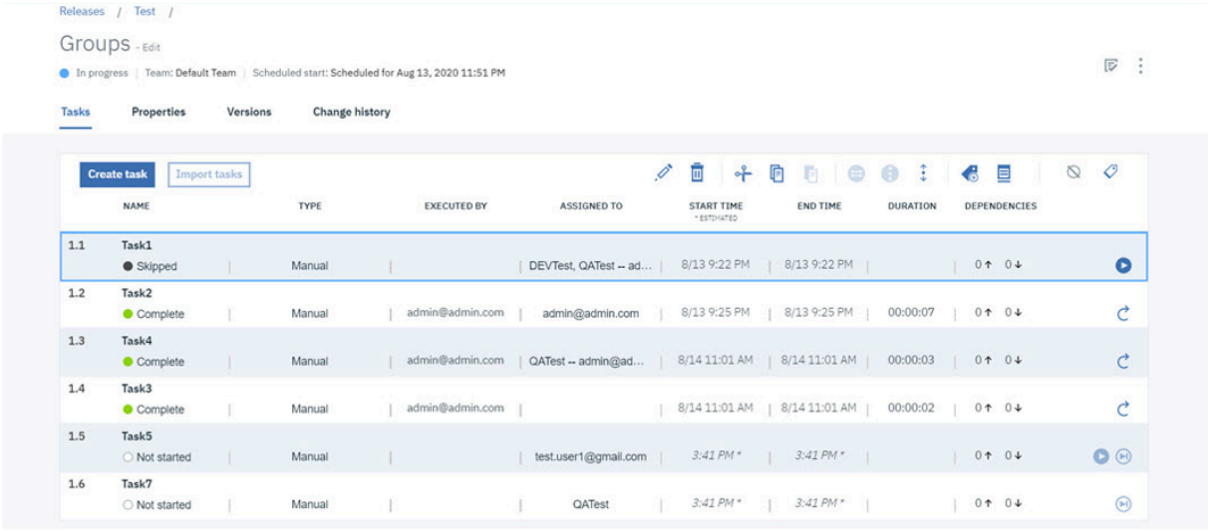


Table 657. Task properties

Property	Description
Name	Task name
Status	Task status: Not started, complete, failed, skipped, not applicable

Table 657. Task properties (continued)

Property	Description
Type	Type of task: Delayed, Deployment Risk Analysis, Email, Header/Note, Jenkins®, Manual, Run another plan, ServiceNow™, HCL DevOps Deploy, and Wait for approval.
Executed by	Person who executes the manual task
Assigned to	Groups and users designated as the task owner
Start time	Start time or expected start time based on scheduled start, or estimated duration of other tasks
End time	Time that the task resolved
Duration	Length of time in minutes from task start to task resolution.
Dependencies	Indicates the number of tasks that are prerequisites for the task, and dependent on the task
Action icons	One or more actions used to change the status of the task during a deployment

Tasks are listed in the order that they run. The order of the tasks can be changed by selecting one or more tasks and dragging to the new location.

At the end of each row are applicable actions associated with the task. Depending on the task you can perform one or more of the following actions.

- Start task ( )
- Skip task ( )
- Re-run task ( )
- Complete task ( )
- Fail task ( )

You can manage the tasks in a deployment plan from the task bar above the table. When a task is selected, the task bar displays and you can perform the following actions:

- Click ( ) to edit the task.
- To copy a task or group, select one or more tasks and click , and then place the cursor where you want to insert the copied task and click to paste the selection.
- To cut a task or group from a deployment plan, select the task and click .
- To delete a task, select the task and click . The task is removed from the deployment plan.
- To create a group of tasks that can run simultaneously, select the tasks and click . The group must consist of at least two tasks.
- To create a group of tasks that are to consecutively run, select the tasks and click . This is the default execution pattern.
- To add or remove tasks that must complete first, select the task and click . In the dialog box, select or type the prerequisite task to be added. Click **X** next to the prerequisite to be removed. When finished, click **Save**.
- To add or remove tags, select the task and click . In the dialog box, type or select tags to add. To remove a tag, click **X** next to the tag to be removed. When finished, click **Save**.
- To view information about when the task started, completed, and errors, click .

Creating tasks

By default, new tasks are inserted at the bottom of the deployment plan. After a task is created, you can move it, or copy it and paste it into another deployment plan. You can also create dependencies with other tasks on page 1848.

About this task

Tasks can be added to a deployment by creating a new task or importing an existing task.

After you save a task, actions are displayed for the task. You use actions to change the task's status during a deployment.

All tasks have the skip action. Other actions, such as the start action, are displayed when the context is appropriate for them.

1. From the **Deployment Plan** details page, click **Create task** to create one or more tasks.
2. Select a task type.

   Note: The availability of task types will depend on your version of HCL DevOps Deploy (Deploy) and integrations.

   Possible task types include:
   - Ansible
   - Delayed
   - Deployment Risk Analysis
   - Email
   - Header / Note
   - Jenkins
   - Manual
   - Run a Shell Script (requires shell script plugin)
   - Run another plan
   - ServiceNow™
   - Deploy (requires Deploy integration)
   - Deploy Status (requires Deploy integration)
   - Wait for approval
3. Provide information in required and optional fields displayed according to the task type selected.
4. Click **Save and create another** to create additional tasks. Click **Save** when finished.

Managing task dependencies

You can make a task a prerequisite for other tasks. If a task is a prerequisite, dependent tasks cannot start, even if they are otherwise eligible, until the prerequisite task is resolved.

About this task

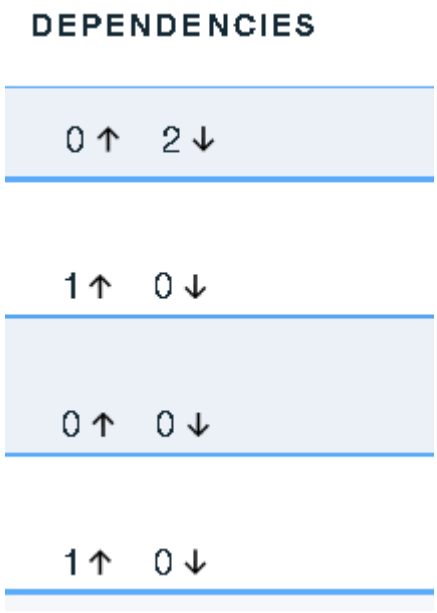
A task can have multiple dependent tasks and multiple prerequisite tasks. You can define dependencies for a task with any other task in the deployment plan. However, you cannot create circular dependencies. You cannot, for example, make a task dependent on a task that itself depends on the first task. By controlling task dependencies, you can ensure that events occur in their expected order.

To make a task a prerequisite for other tasks, complete the following steps:

1. On the **Deployment Plan** details page, select a task or task group, and then click . You can select multiple tasks and groups.
2. In the **Manage prerequisites for selected tasks** dialog box, in the **Prerequisite tasks for selected tasks** drop-down list, select the prerequisite task.
3. Click **Save**.

Result

Task dependencies are shown in the **DEPENDENCIES** column on the **Deployment Plan** details page. Up arrows indicate task prerequisites; down arrows indicate task dependencies. In the following figure, the first task does not have any prerequisites and has two tasks dependent on it. The second task has one prerequisite task and there are no tasks dependent on it.



What to do next

To review or modify dependencies, select the task and then click . Use the **Manage prerequisites for selected tasks** dialog box to modify or remove dependencies.

Creating task groups

You can combine two or more tasks into a task group. When you create a group, you define the whether the group task are executed sequentially or in parallel.

About this task

Tasks can run in a parallel or sequential pattern. When you create the task, you decide the execution pattern. Tasks without dependencies can execute in any order or simultaneously which is *parallel* execution. Tasks that must execute in specific order are *sequential* execution. Tasks are done in list-order starting with the first or top-most task.

You can embed groups within other groups. A sequential-pattern group can be embedded within a parallel-pattern group, and vice versa; however, you cannot embed a sequential-pattern group within another sequential group, or a parallel-pattern group within another parallel group.

1. On the **Deployment Plan** details page, select two or more tasks.
2. Depending on the type of group you want to create, complete one of the following.
   Choose from:
   - To create a parallel group, click the Parallel icon)). If you cannot create a parallel group with the selected tasks, the icon is disabled. For example, you cannot create a parallel group if all the selected tasks are already in a parallel group.
   - To create a sequential group, click the Sequential icon ( ).

   Result
   The group is formed and a group select bar is added to the deployment plan. If you selected discontiguous tasks, the tasks form a contiguous list starting with the topmost selected task.

Managing task groups

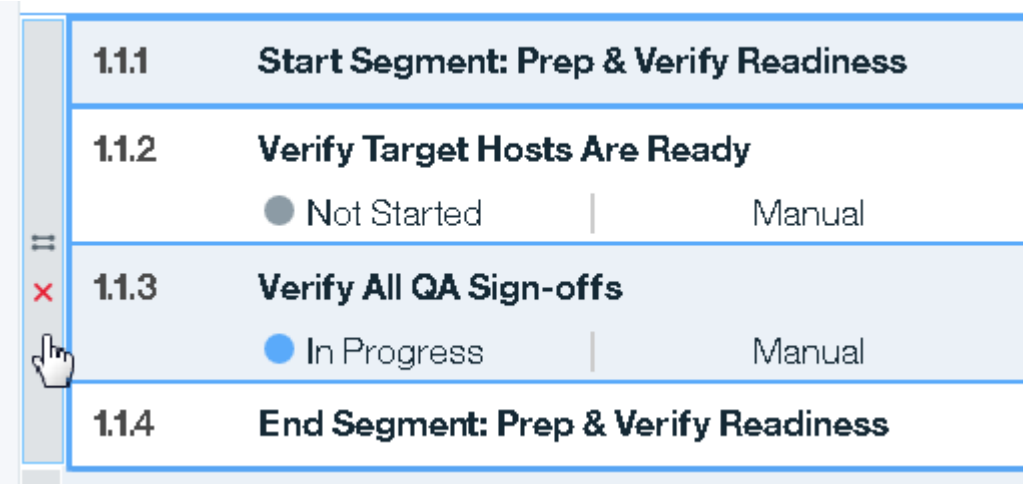
You can combine two or more tasks into a task group. When you create a group, you define the whether the group task are executed sequentially or in parallel.

About this task

You can recognize task groups in the list of task, there is a bar connects the grouped tasks and contains an icon to identify if it is a parallel or sequential group.

- parallel
- sequential

The following image shows a parallel group.



You can manage task groups in several ways:

- To move a group, select the group select bar or click anywhere on the group, and then drag it to a new location.
- To copy a group, select the group and click , and then place the cursor where you want to insert the copied group and click .
- To cut a group, select the group and click .
- To ungroup a group, select the group and click the **Ungroup** icon on the group select bar.
- To delete the tasks in a group, select the group and click . The tasks are removed from the deployment plan.

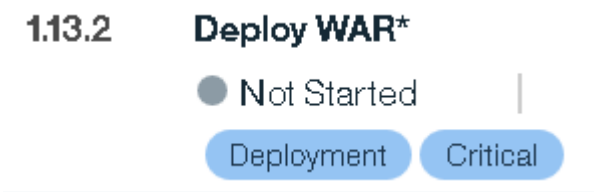
Managing task tags

Tags are organizing elements that you can add to tasks. You can filter deployment plans by tag. For example, during a deployment to a production environment, you might disable tasks with the `DEV_only` tag, which is intended to be used only in development environments.

About this task

To add a tag to a task, complete the following steps:

1. On the **Deployment Plan** details page, select a task or task group, and then click . You can select multiple tasks and groups.
2. In the **Manage Tags for Selected Tasks** dialog box, in the **Tags** drop-down list, select tags. You can create a new tag by typing a name in the list's text box.
3. Click **Save**.
   Tags are displayed on the task rows in the **Deployment Plan** details page. In the following figure, the Deploy WAR task has two tags assigned to it, `Deployment`, and `Critical`.



   The tags used by a deployment plan are displayed on the **Deployment Plan** details page by clicking **Versions**. To render tasks with a specific tag `Not Applicable` for a deployment, clear the tag. Tasks with the `Not Applicable` status cannot be started.

Creating run another plan tasks

A run another plan task runs a deployment for another deployment plan.

About this task

The targeted plan can be created from a template or a tag. If you use a template, the template and the targeted plan must be in the same release as the run another plan task.

To create a run another plan task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create task**. If you want to insert a task at a specific position in the plan, select a task before you use the **Create task** action. The new task is inserted before the selected task.
2. In the **Create task** dialog box, in the **Type** drop-down list, select **Run another plan**.
3. In the **Name** field, enter a name for the task.
4. In the **How do you want to choose another plan** area, select an option.
   If you choose, **By template**, the template and associated deployment plan must be in the same release as the task's parent plan. If you choose, **By tag**, the tagged deployment plan must be in the same release as the task's parent plan as demonstrated in this example on page 1852 of creating a primary and secondary template.
5. In the **Run plans created from this template** list, select a deployment plan template. The template and associated deployment plan must be in the same release as the task's parent plan.
6. In the **Run plans with this tag** list, select the tag associated with the template to the task.
   You can select multiple tags. To create a tag, type the tag name in text box.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When the task runs, a deployment starts for the deployment plan that was created from the selected template or tagged plan. If multiple plans in the release event are eligible, deployments start for all of them. While the other deployments run, status information is provided by the run another plan task. Expand the task to see the status information. You can open the other deployments by using links on the task. While the other deployments run, the run another plan task has a status of **In Progress**.

Creating run another plan tasks: Example

A run another plan task runs a deployment for another deployment plan.

About this task

This example is a targeted plan that is created from a tag. The tagged deployment plan must be in the same release as the task's parent plan.

To create a targeted plan that is created from a tag, complete the following steps:

1. Create the primary template.
    a. On the **Releases** page, click **Templates > Create template**.
    b. In the **Plan template name** field, enter the name for the primary template.
    Accept the default values for the remaining fields.
    c. Click **Save**.
    d. Click the name of the primary template.
2. Referring to the Creating tasks on page 1847 page, create independent and dependent tasks for the primary template by using field values that are detailed in the following tables for each task:

Table 658. Manual task

Create task field	Value
Type	Select **Manual**.
Name	Enter the name for the manual task.
Description	Default value.
Duration (minutes)	Default value.
Tags	Default value.
Assigned groups and users	Default value.

Table 659. Run another plan task

Create task field	Value
Type	Select **Run another plan**.
Name	Enter the name for the secondary task.
How do you want to choose another plan?	**By tag**
Run plans with this tag	Select a tag or create a new tag by typing a name in the field. The primary deployment plan will find any other plan with this tag name when it runs.
Tags	Default value.

3. Create the secondary template.
    a. On the **Releases** page, click **Templates > Create template**.
    b. In the **Plan template name** field, enter the name for the secondary template.
    Accept the default values for the remaining fields.

c. Click **Save**.

d. Click the name of the secondary template.

4. Referring to the Creating tasks on page 1847 page, create independent and dependent tasks for the secondary template by using field values that are detailed in the following tables for each task:

Table 660. Wait for approval task

Create task field	Value
Type	**Wait for approval** or any task that starts automatically.
Name	Enter the name for the task.
Approval prompt	Default value or enter a prompt message.
Approvers	Select the name of the approver. You can select multiple approvers.
Required Approver	Select **Any approver can approve** or **All approvers must approve**

Table 661. Manual task

Create task field	Value
Type	**Manual**
Name	Enter the name for the task.
Description	Default value.
Duration (minutes)	Default value.
Tags	Default value.
Assigned groups and users	Default value.

5. Referring to the Creating releases on page 1833 page, create a release with the primary and secondary templates by using field values that are detailed in the following table:

Table 662. Create release

Create release field	Value
Name	Enter the name for the release.
Tags	Select the name for the required tag.
Description	Default value.
Deployment plan templates	Select the name of the primary template that you created.

Create release field	Value
Start time	Default value.
End time	Default value.
Team	Default value.

6. On the **Releases** page, click the name of the release.
7. Click the primary plan to display the independent task and dependent task with a tag in the template.
8. At the top of the page, click the link to go back to the release.
9. Create the deployment plan.
    a. Click **Create deployment plan**.
    b. In the **Deployment plan template** list, select the name of the secondary template that you created.
    c. In the **Tags** field, select the name for the required tag.
    d. Click **Save**.

    Note: You can create one deployment plan as a primary linked to many secondary deployment plans. Deployment is executed only from primary deployment plan.

10. Click the primary plan to display the independent and dependent tasks.
    The dependent task is now ready to run with the independent task being a manual type.

    a. On the manual task, click **Start task** to start the task.

    b. On the manual task, click **Approve task** to complete the task.
    The manual task displays the status as `Complete`, and the dependent task displays `In progress`.

    c. On the dependent task, click the arrow to the left of **In progress**.

    d. Click **Go to** to go to the dependent task.

    e. On the "Wait for approval" task, click **Approve task** to approve the task.

    f. On the "Manual" task, click **Start task** to start the task.

    g. On the "Manual" task, click **Complete task** to complete the task.

    h. At the top of the page, click the link to go back to the release.

The **Manual** and **Run another plan** tasks are both displayed as completed in the secondary plan.

i. At the top of the page, click the link to go back to the primary plan to display the plans in the release that are completed.

Creating delay tasks

Delayed-type tasks represent milestones or critical events during a deployment. With delayed tasks, you can ensure that important tasks start at the expected time. Typically, delayed tasks are prerequisites for other important tasks.

About this task

A delayed task starts as soon as it is eligible to run, and it finishes at a user-specified time. A started delayed-type task has a status of `In Progress` until it reaches its planned-for time, when it then changes its status to `Complete`. When a delayed tasks completes, auto tasks that are dependent on it start running.

Complete the following tasks to create a delayed task:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Delayed**.
3. In the **Name** field, enter a name for the task.
4. In the **Delay type** field, select the method used to determine the length of the delay.
   If you select **Delay by selecting a time and time zone**, use the **Delay until time** and **Time zone** fields to specify when the task will complete. If you select **Delay by using a date and time property**, use the **Delay property** field to enter a property name. The property value can be set at run-time when the deployment starts. Refer to the Properties reference for information about properties:

   ```
   /docs/services/UCCR/UCCR_property_ref.html#property_overview
   ```

   Otherwise, if you select **Delay by selecting a number of minutes**, use the **Minutes** field and enter the number of minutes between tasks.
5. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating email tasks

An email task sends an email message when the task runs. You specify the email's recipients and message when you create the task. Email tasks are auto tasks and run as soon as they are eligible.

Before you begin

Before using email tasks in your deployments, setup an email server. on page 1800

About this task

To create an email task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Email**.
3. In the **Name** field, enter a name for the task.
4. In the **Recipients** field, enter or select the email's recipient.
   The list of available recipients includes the users and groups who are members the team that manages the task. You can also type the email addresses of non-team members. You can specify multiple recipients.
5. In the **Email subject** field, enter the email topic.
   You can use properties in this field.
6. In the **Email message** field, enter or paste the email message.
   You can use properties in this field. For example, you might notify email recipients that a release is started by using a message similar to this one, "The `` `${sys:release.name}` `` release has started." See the Properties reference for information about properties on page 1871.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When the task runs, the recipients receive an email with the subject you specified when you created the task.

Creating header/note tasks

Header tasks represent organization elements that you can add to deployment plans. If you create a task group, you might identify the group with a header task. Header tasks can have dependencies like any other task.

About this task

You can organize task groups with header/note tasks. You can create a note by adding information about a group or task. When you import a deployment plan from IBM® DevOps Release, segment tasks are bracketed by note-type Start Segment and End Segment tasks. Segment dependencies are represented by dependencies to End Segment tasks.

To create a header task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Header / Note**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter or paste a description.
   You might enter a note, a reminder, or other instructions in the field.
5. Click **Save**.

Result

The task is inserted into the deployment plan.

Creating Jenkins (Legacy) tasks

A Jenkins (Legacy) task runs a job in your integrated Jenkins instance.

Before you begin

Before using Jenkins (Legacy) tasks in your deployments, add the plugin to your Jenkins instance and configure an integration. on page 1886 After integrating Measure to Jenkins, you can add Jenkins jobs to you pipelines and releases.

About this task

To create a Jenkins (Legacy) task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Jenkins (Legacy)**.
3. In the **Name** field, enter a name for the task.
4. In the **Jenkins Job or Pipeline** field, select a job or pipeline.
5. In the **Duration (minutes)** field, enter an estimate for the length of the task.
6. In the **Tags** list, select an item to attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating manual tasks

Manual tasks represent some activity that is associated with a software release. Manual tasks have a start point, end point, and a measurable duration.

About this task

To create a manual task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Manual**.
3. In the **Name** field, enter a name for the task.
4. In the **Duration (minutes)** field, enter the number of minutes that you expect the task to run until it is completed.
   The estimated duration is used to calculate expected deployment times.
5. In the **Tags** list, select an item to attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.

6. In the **Assigned groups and users** list, select the task owner.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

Once the manual task is executed, you can select the task on the deployment plan details page on page 1845, click to display the **Edit task** window, and view who executed the task in the **Executed By** field.

Creating run shell script tasks

Shell script tasks are used to execute a sequence of commands.

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create task** dialog box, in the **Type** list, select **Run a Shell Script Task**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter or paste a description.
   You might enter a note, a reminder, or other instructions in the field.
5. In the **Shell Script** field, enter or paste the shell script code. An example shell script code is the following:

```
echo "Hello World"
sleep 2
"Do some stuff..."
sleep 2
echo "Task finished"
```

6. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating ServiceNow tasks

A ServiceNow™ task can create and update ServiceNow™ change requests.

Before you begin

ServiceNow task prerequisites:

- ServiceNow tasks work with Internet-accessible instances of ServiceNow, such as `my_instance.service-now.com`.
- ServiceNow tasks work with the Jakarta® release of ServiceNow, and references the current API version.
- The user supplying credentials for the task must have access to the ServiceNow API, and permission to see and modify all change request fields. At a minimum, the user must be in a ServiceNow role that has the Create Change permission.

About this task

A ServiceNow task can perform one of the following actions:

- **Create** creates a ServiceNow change request. You can set the values for all request fields, and pass the request ID to update, and wait tasks.
- **Update** can update the values for any request field.
- **Wait** waits for a particular value in a specific field. The field can be any field on the change request.

A ServiceNow task can include ServiceNow properties and Measure properties. ServiceNow properties have the following syntax: `propertyName=value`.

This short video describes how to use ServiceNow with Measure.

To create a ServiceNow task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **ServiceNow**.
3. In the **Name** field, enter a name for the task.
4. In the **Tags** list, attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.
5. In the **Connection settings** area, enter the URL of the ServiceNow instance, and the user ID, and password of the ServiceNow user.
6. In the **Action** list, select the action type for the task.
   The action types are defined in this list:
   - The **create** action creates a ServiceNow change request with the properties defined in the task.
   - The **update** action modifies properties in change request.
   - The **wait** action waits for a specific value in a change request field.
7. Complete the action-specific parameters, as described in the following tables:

Table 663. Create action

Create action	Description
Request type	You can use the ServiceNow default types or a user-created type. Default types are normal, emergency, and standard. Normal requests pass through all stages

Create action	Description
	of the ServiceNow state model. Standard types are pre-authorized and skip the Assess and Authorize states. Emergency types skip the Assess state.
Short description	The text you enter is displayed in the Short description field of the ServiceNow change request.
Assignment group	The ServiceNow agent or group responsible for resolving the change request. Possible assignees include agents and groups available to the ServiceNow instance.
Additional properties	The ServiceNow properties set by this task. Each property must be on its own line. For example, a simple request might include the following fragments: `approval=requested`.
Output property	Measure property that contains the change request ID. For example, if you specify `service_now_stage_ticket_id`, you can use the property with update and wait actions to identify the ticket.

Table 664. Update actions

Update action	Description
System ID	The ID of the ServiceNow change request. If you capture the ID with a create action **Output property** field, you can use it here. For example, if you specify `service_now_stage_ticket_id` in the **Output property** field of a create action, you can use the ID by specifying `${service_now_stage_ticket_id}`. See the Properties reference for information about properties on page 1871.
Additional properties	The ServiceNow properties set by this task. Each property must be on its own line. For example, to update the state of the change request, you can specify or `state=-1`.

Table 665. Wait actions

Wait action	Description
System ID	The ID of the ServiceNow change request. If you capture the ID with a create action **Output property**

Wait action	Description
	field, you can use it here. For example, if you specify `` `service_now_stage_ticket_id` `` in the **Output property** field of a create action, you can use the ID by specifying `` `${service_now_stage_ticket_id}` ``.
Field	The name of the affected field in the ServiceNow change request. The value in the field is checked for updates by the interval defined by the Check request interval field. For example, if you are checking the **approval** field, you might specify `` `approval` ``.
Value	The value expected in the ServiceNow field as defined by the **Field** property. For example, if you are checking the **approval** field, you might specify `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` ``, the task completes successfully.
Check request interval	The value defines how frequently ServiceNow is checked for updates. The value represents minutes. The default value is `1`, which mean ServiceNow is polled every minute.

Result

When the task starts, it has the status of **In progress**. The status changes to **Complete** when the related ServiceNow action is resolved.

What to do next

The ServiceNow change request life cycle is referred to as the state model. As a change request is used, it progresses through the various states of model. The state of a closed request is 'closed', for example.

In Measure, ServiceNow tasks are frequently used to change the state of a request. To modify a request state, you use the ServiceNow `state` property. For example, to change the state of a request to Review, you can specify **`state=0`** in the **Additional properties** field of an Update action.

The following table provides the values for the default states for the normal request type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 666. ServiceNow normal request state model

State	Value
New	-5
Assess	-4
Authorize	-3

Table 666. ServiceNow normal request state model (continued)

State	Value
Scheduled	-2
Implement	-1
Review	0
Closed	3
Canceled	4

Creating ServiceNow task using Service Now plug-in

A ServiceNow™ task can create and update ServiceNow change requests and the change tasks within the change request. You can also create a ServiceNow - Wait Change Request task which will wait for the specified value to appear in a specific field of a change request in ServiceNow.

Before you begin

You must have completed the following tasks:

- Ensured that you have internet-accessible instances of ServiceNow, such as `my_instance.service-now.com`.
- Created an integration with ServiceNow plug-in. For more information, refer to ServiceNow plug-in documentation.

Note: Ensure that the credentials used for the plug-in integration must have access to the ServiceNow API and permission to see and modify all change request fields. At a minimum, the user must be in a ServiceNow role that has the Create Change permission.

About this task

You can create the following automation tasks with which you can perform the actions in ServiceNow directly from Measure:

- **ServiceNow - Create Change Request and Change Task**

  This task creates a change request, change task or both in ServiceNow.

- **ServiceNow - Update Change Request and Change Task**

  This task updates the value in any change request, change task, or both in ServiceNow.

- **ServiceNow - Wait Change Request**

  This task waits for specified value to appear in a specified field of a change request in ServiceNow. The task will be completed when the provided value appears in the ServiceNow change request. The field can be any field of a change request.

- **ServiceNow - Wait Change Task**

  This task waits for specified value to appear in a specified field of a change task in ServiceNow. The task will be completed when the provided value appears in the ServiceNow change task. The field can be any field of a change task.

1. Click **Releases** on the left navigation pane.
   Result
   The **Releases** page is displayed.
2. Click any deployment plan.
   Result
   The **Deployment Plan** details page is displayed.
3. Click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
   Result
   The **Create Task** dialog is displayed.
4. In the **Type** list, select any one of the following options based on the action that you want to perform in ServiceNow:
   - **ServiceNow - Create Change Request and Change Task**
   - **ServiceNow - Update Change Request and Change Task**
   - **ServiceNow - Wait Change Request**
   - **ServiceNow - Wait Change Task**
5. In the **Name** field, enter a name for the task.
6. In the **Description** field, enter the description for the task.
7. In the **Integration** field, select the integration which you have created for ServiceNow.
8. Based on the task type you selected, enter the details in the task specific parameters, as described in the following tables:

Table 667. ServiceNow - Create Change Request and Change Task

ServiceNow - Create Change Request and Change Task type	Description
Short description for change request	Enter the text that you want to display in the Short description field of the ServiceNow change request.
Request type	Enter the ServiceNow request type. you can use the ServiceNow default types or a user-created type. Default types are normal, emergency, and standard. Normal requests pass through all stages of the ServiceNow state model. Standard types are pre-authorized and skip the **Assess** and **Authorize** states. Emergency types skip the **Assess** state.

ServiceNow - Create Change Request and Change Task type	Description
Assignment group for change request	Enter the ServiceNow agent or group responsible for resolving the change request. Possible assignees include agents and groups available to the ServiceNow instance.
Additional properties for change request	Enter the ServiceNow properties that you want to update by this task while creating the change request. For example, a simple create change request might include the following fragments: {"approval:requested"}.
Create change task	Enter the ServiceNow properties that you want to update by this task while creating the change task within the change request. You can create multiple change task by separating each change task properties object by a comma. For example, a simple create change task might include the following fragments:[{"short_description":"createtask","change_task_type":"planning","description":"changetask"}]
Output property	Enter the system ID of the created change request.

Table 668. ServiceNow - Update Change Request and Change Task

ServiceNow - Update Change Request and Change Task type	Description
Change Request Number	Select the Change Request Number of the ServiceNow change request that you want to update. See the Properties reference for information about properties on page 1871.
Change request properties	Enter the ServiceNow properties that you want to update by this task while updating the change request. For example, to update the state of the change request, you can specify or `{"state:-1"}`.
Update change task	Enter the ServiceNow properties that you want to update by this task while updating the change task within the change request. You can update multiple change task by separating each change task properties object by a comma. For example, to update the state of the change task, you can specify or `[{"state:3"}]`.

Table 669. ServiceNow - Wait Change Request

ServiceNow - Wait Change Request type	Description
Change Request Number	Select the Change Request Number of the ServiceNow change request. See the Properties reference for information about properties on page 1871.
Field	Enter the name of the field for which the task should check for a match in the ServiceNow change request. The value in the field is checked for updates in a regular interval. For example, if you want to check for a match in the **approval** field, you might specify `` `approval` ``.
Value	Enter the value expected in the ServiceNow field defined in the **Field** property. For example, if you are checking the **approval** field, you can specify a value `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` `` in ServiceNow, the task completes successfully.

Table 670. ServiceNow - Wait Change Task

ServiceNow - Wait Change task type	Description
Change task system id from property reference.	Enter the system ID of the change task.
Change Request Number	Select the Change Request Number of the ServiceNow change request. See the Properties reference for information about properties on page 1871.
Change request system id from property reference.	Enter the system ID of the change request.
Field	Enter the name of the field for which the task should check for a match in the ServiceNow change request. The value in the field is checked for updates in a regular interval. For example, if you want to check for a match in the **approval** field, you might specify `` `approval` ``.
Value	Enter the value expected in the ServiceNow field defined in the **Field** property. For example, if you are checking the **approval** field, you can specify a value `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` `` in ServiceNow, the task completes successfully.

9. Click **Save** or **Save and create another** to add the task or add the task and create another task respectively.

Result

When the task starts, it will have the status of **In progress**. The status changes to **Complete** when the action mentioned in the task is performed in ServiceNow.

Example

Additional Information

The ServiceNow change request life cycle is referred to as the state model. As a change request is used, it progresses through the various states of model. For example, the state of a closed request is 'closed'.

In Measure, ServiceNow tasks are frequently used to change the state of a change request and change task.

To modify a request state, you use the ServiceNow `state` property. For example, to change the state of a change request to **Review**, you can specify "state:0" in the **Change request properties field** of a **ServiceNow - Update Change Request and Change Task** type and to change the state of a change task to **In progress**, you can specify "state:2" in the **Update change task** field of a **ServiceNow - Update Change Request and Change Task** type.

The following table provides the values for the default states for the normal change request type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 671. ServiceNow normal change request state model

State	Value
New	-5
Assess	-4
Authorize	-3
Scheduled	-2
Implement	-1
Review	0
Closed	3
Canceled	4

The following table provides the values for the default states for the normal change task type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 672. ServiceNow normal change task state model

State	Value
pending	-5
open	1
in progress	2
closed	3
canceled	4

Creating DevOps Deploy tasks

Use DevOps Deploy tasks to run DevOps Deploy applications.

Before you begin

Before you can use DevOps Deploy tasks in pipelines and deployment plans, you must first configure DevOps Deploy integration on page 1885.

The applications that are available to a deployment plan depend on the team that is assigned to the plan. The applications that are managed by the team in DevOps Deploy are also available in Measure.

About this task

To create this task type, specify the application's process, version, and environment. You can set the version and environment at design time or select them at run time. When you run this task, the application runs in the integrated DevOps Deploy instance. During deployments, DevOps Deploy tasks start automatically when they become eligible to run.

To create a DevOps Deploy task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **DevOps Deploy**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, provide the description of the task.
5. In the **Integration** drop-down list, select the plugin integration.
6. Select the **Assigned group or user** from the drop-down list.
7. In the **Application** drop-down list, select an application.
8. In the **Process** drop-down list, select or enter an application process.

Processes that belong to the selected DevOps Deploy application are available.

9. In the **Version** drop-down list, select an application version.
   Versions refer to DevOps Deploy application snapshots. Versions that belong to the selected application are available. To postpone selecting a version, select **Use Version Tab**. If the application process does not require a version, select **No version items exist**. You might select this last option if you are running a configuration-type process that does not require components.
10. In the **Environment** drop-down list, select or enter an application environment.
    Environments that belong to the selected DevOps Deploy application are available. To postpone selecting an environment until you are ready to run the deployment, click **Versions** on the **Deployment plan** details page.
11. In the **Deploy Request Properties** box, enter the properties that you want to pass to the IBM DevOps Deploy (Legacy) application.
    Properties are name-value pairs. You can enter multiple properties. Put each property on a separate line. The following graphic provides an example of property definitions:
    Figure 4. property definitions

12. Select the snapshot display from the **On Failure** and **On Success** drop-down list.
13. Click **Save**.
    Result
    The task is inserted into the deployment plan.

Results

After the task is created, the plan's **Versions** page is updated with information about the application. If you selected **Only Changed Versions** for the application environment and version on the **Create Task** dialog box, use the **Versions** page to set those options before running the deployment.

Creating Ansible tasks

Create an Ansible® task.

Before you begin

In your Ansible® instance, create a user token. You use the token to authenticate your Measure user to Ansible®.

About this task

Create an Ansible® task to manage Ansible® jobs.

To create an Ansible® task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Ansible**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter a name for the task.
5. In the **Instance URL** field, enter the location of your Ansible® installation..
6. In the **Access Token** field, paste the token that you generated in your Ansible® instance.
7. In the **Job ID** field, enter the Ansible® job identification number..
8. **Optional:** In the **Extra Variables** box, enter any parameters that you want to pass to the Ansible® job.
   The parameter format is JSON. For example:

   ```
   {
    "environment": "DEV"
   }
   ```

   example,
9. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When you run the task, a webhook is created that uses the Ansible® API to run the job. While the job is running, you can accesses Ansible® logs by using a link in the deployment plan. When the Ansible® job is finished, the Measure task status is updated.

Creating wait for approval tasks

A Wait for approval task pauses a deployment until a designated user provides approval. Email notifications are sent to users when the task starts.

Before you begin

Before using email tasks in your deployments, setup an email server. on page 1800

About this task

To create a wait for approval task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Wait for approval**.
3. In the **Name** field, enter a name for the task.
4. In the **Approval prompt** field, enter the text that you want to include in the email message.

The text is displayed in the **Task Details** section of the email.

5. In the **Approvers** list, select the users or groups designated to approve the task.
   Team members and groups are available to choose.
6. In the **Required approver** area, specify whether all the approvers must approve the task, or a single approver is sufficient.
7. Click **Save**.
   The task is inserted into the deployment plan.

Results

When the task runs, email notifications are sent to the selected approvers. The deployment plan cannot progress until the task is approved or skipped. Approval is granted by completing the task in the deployment plan.

Note: For information on pipeline notifications, see this page on page 3149.

Approving wait for approval task

You can approve the wait for approval task using the received email notifications.

About this task

Designated approval user receives the email notification for approving the wait for approval tasks. The email notification contains the task information for which the wait for approval task needs to be approved.

You can click **Approve** or **Reject** for approving or rejecting the wait for approval task.

By default, designated users can approve the task or share the email notification to any user for approving. You can prevent the latter functionality by enabling the `authorized-approval` feature flag.

To activate this security feature, enter the feature name `authorized-approval` under `featureFlags` . After enabling `authorized-approval` feature flag , only designated users can approve or reject the wait for the approval task. To approve or reject, the user must login Measure. Not designated users are not authorized to approve or reject the wait for approval task using the email notification.

Using properties

A deployment plan can contain user-defined properties. Properties values can be defined at design- or run-time.

Tasks can reference properties defined in their parent deployment plan. In addition to the properties that you create, system properties are available to all plans.

Creating properties

Creating properties

About this task

You define properties with the **Deployment Plan** details page by clicking **Properties**. You can create a property before you use it, or you can create a property by first referencing it in a task, and then later defining its attributes. If you create a property by reference, it appears as text-type property on the **Properties** tab.

To create a property, complete the following steps:

1. On the **Deployment Plan** details page, click **Properties**, and then click **Add**.
2. On the **Add property** window, in the **Name** field, enter the name. Property names cannot contain white spaces, `{}`, or `:`.
3. In the **Label** field, enter the label that appears in the user interface.
4. In the **Type** list, select the property type.
   The available types are described in this list:
   - `Text`
   - `Textarea`
   - `Password`
   - `Select`
   - `Boolean`
   - `Date and time`. Values are replaced by a string with the following format `day, month, year, time`. For example, `Thursday, January 1st 1970, 12:00 am UTC`.

     Note: When a date and time property is used in a ServiceNow® task, the value is replaced with the date and time format expected by ServiceNow®.

5. In the **Value** field, you can enter an initial value for the property.
6. In the **Default Value** field, enter a value for the property.
   This value is used unless the user sets the value. You can leave this field blank.
7. If you want users to supply a value for the property when they create a release, check **Prompt for this property at release creation at the specified sequence position**.
8. In the **Display sequence position** list, select a position for the property.
   This selected value affects the property's position on the **Properties** tab
9. Click **Save**.
   Result
   The property is displayed on the deployment plan **Properties** page.

What to do next

After the release is created, you can add deployment plans, events, and tags to it.

Using properties

You can use properties in deployment plans, REST API endpoints, and external systems, such as ServiceNow®.

Property examples

Use the following syntax to reference a property: `` `${propertyName}` ``. To escape the syntax pattern, use two dollar-sign symbols, `$${...}`. You set property values with the deployment plan's **Properties** page, REST API endpoints, or external systems, such as ServiceNow®.

The following table describes how to properties with task types.

Task type	Usage
ServiceNow	You can use the ServiceNow on page 1863 **Output property** field in a create change request action to store the change request ID in a property, for example, `'service_now_prod_ticket_id'`. Then, with another task, update the change request by referencing the property in the **System ID** field like this, `'${service_now_prod_ticket_id}'`.
Email	You can reference a system property in the email task on page 1856 **Subject** field, for example, `'${sys:release.name}' is complete!`.
Delayed	You can reference a delay task on page 1856 property in the **Delay property** field, for example, `'${prodTarget}'`. In this example, the task delays until the value in the `'${prodTarget}'` property is reached.

System properties

System properties enable you to access information about the parent release of a deployment plan. System properties can be referenced by any property-enabled task that is part of a release.

The following table contains the available system properties:

Table 673.

Property	Description
`${sys:release.name}`	The value in the release's **Name** field.
`${sys:release.description}`	Contains the content of the release's **Description** field.
`${sys:release.start}`	Release's **Start time**.
`$${sys:release.end}`	Contains the content of the release's **End time** field.

In the **Email message** field of an email-type task, you can reference system properties like this example:

```
Deployment for release `${sys:release.name}` - `${sys:release.description}` is complete.
```

You can reference a system property in a ServiceNow task's **Short description** field like this example:

`Staging deployment `${sys:release.name}``.

Extending Product Functions

Learn how to extend the function of Velocity Measure

Measure is designed to be easily extended. You can enhance existing functions or add entirely new ones by configuring plugin integrations. Plugins are software add-ons that extend Measure functionality by integrating with external tools. In general, each plugin manages one external application. The HCL DevOps Deploy (Deploy) plugin, for example, enables you to run Deploy applications.

If you cannot find a plugin for your environment, you can create plugins by using the plugin software development kit (SDK) on page 1884.

Measure supports several plugin types:

- Value stream plugins are supported by default. To create an integration with these plugins, you simply configure an integration in the value stream JSON file.
- Quality data plugins integrate test and metrics data into reports and value streams on page 1893. To create an integration with these plugins, you use external clients to post data to Measure API endpoints.
- Dedicated plugins provide external data to value streams, pipelines, and releases. Plugins of this type include: Deploy on page 1885, and Jenkins on page 1886. Use the Integrations page to configure integrations for these plugins.
- Task-based integrations provide external data to deployment plans. Use ServiceNow tasks to manage ServiceNow change requests on page 1887.

Introducing plugins

Extend Measure by configuring plugin integrations to external tools.

Measure uses a containerized microservices architecture. This means that plugins are not run directly from Measure but from containerized instances managed by a containerization platform. Runtime instances are created from images on the Docker Hub™ repository. Measure retrieves images from this repository and creates instances as needed.

For Kubernetes-based installations such as OpenShift™, Measure uses Kubernetes™ as its containerization platform. An Argo Workflow Engine manages plugin workflow. A Kubernetes pod starts when a plugin task is requested and sends the results to Measure. When complete, the container containing the plugin ends.

Plugins are categorized by data collection and communication methods. Generally, plugins are designed to use one of the following communication methods.

- **Webhook**. These plugins use webhooks to communicate with a defined Measure API endpoint. The webhook is used to trigger data collection events. Examples include the AppScan™, and SonarQube plugins.
- **Poller**. These plugins are based on an event defined by the plugin. Queries are performed to determine when to send and update data from the external service. Examples include the GitHub™, and Rally plugins.
- **Parser**. These plugins integrate functions to parse a specific file type and create a metric document that Measure can display with other value stream management and portfolio views. Examples include the JUnit, and OneTest plugins.

Available plugins are stored at HCL plugin database.

If you cannot find a plugin for your environment, you can create plugins by using the plugin software development kit (SDK) on page 1884.

If a metrics plugin is not available for your tool, you can upload custom metrics using Measure API endpoints.

To use a plugin, you configure an integration. "Integration" refers to a user-configured instance. You might use the GitHub plugin to configure an integration with the Service A repository, and then configure another integration with your Service B repository. Although both integrations use the GitHub plugin, each integration provides a unique set of data to Measure. You manage each integration separately.

Some integrations are termed "native integrations," and are not technically plugin derived. Native integrations are used with deployment plans by defining tasks of a specific type on page 1885. You can use DevOps Deploy data with your deployments and pipelines, for example, by defining DevOps Deploy-type tasks.

Some plugins might have a corresponding plugin in the external tool. To integrate Jenkins, for example, you install the " Measure" plugin into Jenkins™ using the Jenkins plugin manager. Then you configure a corresponding Jenkins integration in Measures. The two integrations operate in tandem.

Preventing plugins installs in unsupported server versions

Plugin versions incompatible with Measure are prevented from installing.

About this task

Based on the your installed version of Measure, plugins not compatible with that version are prevented from installing in the product. If the installed version of Measure is equal or greater than the product version that the plugin supports, the installation of the plugin is possible.

Notifications for incompatible plugins versions prevented from installation are displayed in each of the following three scenarios:

Scenario 1: Installing plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Plugins** tab.

3. Locate the required plugin in the list and click **Install**.
4. The error **Plugin <name> is only supported on** Velocity **versions equal to or greater than x.x.x** is displayed on the right side of the **Status** field for that plugin.

Scenario 2: Loading plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Plugins** tab.
3. Click **Load plugin**.
4. In the **Load a custom plugin** window, click in the **Plugin Image** field and enter the required plugin.
5. Click **Submit**.
6. The error **Plugin <name> is only supported on** Velocity **versions equal to or greater than x.x.x** is displayed in the **Load a custom plugin** window.

Scenario 3: Upgrading plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Integrations** tab.
3. Locate the required plugin and then click on the overflow menu and select **Upgrade**.
4. In the **Status** field for the plugin, **Upgrade failed** is displayed.

Creating plugin integrations

Defining integration allows you to communicate with external tools.

Before you can create an integration, ensure you have the user credentials and connection information required by the external tool, such as URLs, and API endpoints. To create an integration with HCL DevOps Deploy (Deploy), for example, you need a user token generated by Deploy, and the URL where the server is located. Ensure that you have all the necessary connection information before starting.

Credentials and other connection properties vary among individual plugins. You can find property descriptions and use cases for all plugins on the product website.

A plugin is a software add-on designed to communicate with external services and applications. Each plugin defines an expected record type and communication method. Communication can be uni-directional or bi-directional. To use a plugin, you must install and configure an integration.

To view the plugin in Measure, on the **Home** page, click **Settings > Integrations**.

The **Integrations** page has three tabs:

1. Integrations on page 1877
2. Installed Plugins on page 1877
3. Available Plugins on page 1878

Integrations

The **Integrations** tab lists all plugins that are configured for integration. Click the right-angle bracket to view the details of the configured plugin. All integrations are listed regardless of how they are created.

You can use search to find configured plugin integrations. The search query finds matches in the **Name** and **Type** fields. On the **Integrations** tab, you can sort by the **Name**, **Status**, **Type**, and **Last sync** fields in ascending or descending order by clicking a table header.

Click **Show Archived** to see the archived plugin integrations. The following table details the fields on the **Integrations** tab:

Field	Description
Name	The name of the plugin. Click the right-angle bracket to view details of the plugin integration. Click to view plugin documentation.
Status	The working status of the configured plugin integration. Possible values are **Online** and **Failed**. The **Online** status is displayed for working plugin integrations. The **Failed** status is displayed for plugin integrations that are not working. Disrupted communications or inaccurate integration configurations can cause the integration to fail.
Type	The type of the plugin.
Last sync	The last plugin synchronization time.
Disable/Enable	Set to disable or enable the plugin integration.
	Click to **Edit**, **Upgrade/Downgrade**, **View logs**, and **Archive/Delete** plugin integrations.

Installed Plugins

The **Installed Plugins** tab displays all plugins that are installed by default with the Measure installation. The installed plugins are ready to configure for integration.

You can search for installed plugins. The search query finds the matches in the **Name** field. On the **Installed Plugins** tab, you can sort the **Name** field in ascending or descending order by clicking the header.

The following table details the fields on the **Installed Plugins** tab:

Field	Description
Name	The name of the plugin. Click to view all installed versions of the plugin. Click to view plugin documentation.
Version	The version of the installed plugin.
Image	The image of the plugin.
Release Date	The release date of the installed plugin.
Actions	Click **Add integration** and provide details to configure the integration. All the integrations are enabled by default. To create a disabled plugin integration perform the following the steps: • In the **Add integration** window, click **Show hidden properties**, and then clear **Enabled** to create a disabled plugin integration. Click and select **Delete** to delete the installed plugin.

The latest version for Parser plugins are installed with the **Non-Configurable** status.

Available Plugins

The **Available Plugins** tab displays all supported plugins for Measure.

You can search for the supported plugins. The search query finds matches in the **Name** field. On the **Available Plugins** tab, you can sort the **Name** field in ascending or descending order by clicking on the header.

The following table details the fields on the **Available Plugins** tab:

Field	Description
Name	The name of the plugin. Click to view all versions of the plugin. Click to view plugin documentation.
Version	The version of the available plugin.
Image	The image of the plugin.
Release Date	The release date of the available plugin.

Field	Description
Actions	Click **Install** to install the required version of the plugin.

Advanced usage

You can configure integrations for some plugin types by adding an integration definition to a value stream JSON file on page 3078. Using a text editor, define an integration object in the `integrations` array of the `value_stream_name-vsm.json` file. The following code example illustrates a Jira integration.

```
{
     "_id": "velocity-generated-id",
     "type": "ucv-ext-jira",
     "tenant_id": "velocity-generated-id",
     "name": "my-JIRA",
     "disabled": false,
     "properties": {
       "baseUrl": "http://my-jira-url",
       "username": "admin",
       "password": "****",
       "jiraProjects": [
         "My-Project"
       ]
     }
   },
```

Integrations created by the value stream method are displayed on the **Integrations** page along with integrations created by the other methods.

Additional Measure product features requiring you to perform plugin installation before use include the following:

1. Handlebars Reporter
2. Bottleneck detection
3. Parser

The integration is listed on the **Integrations** page. The **Integrations** page is where you manage integrations. on page 1881

Integrations start providing data when successfully configured. For some integration types, such as quality-data types on page 3165, you can start using their data without additional configuration. For example, you can create dashboard charts that contain quality data when the integration is added. For example, JUnit is a quality-type plugin.

Other integration types require additional configuration before you can start using the associated data. Native integrations require that tasks are added to deployment plans. on page 1885 Value stream integrations, such as GitHub™, require a reference in the value stream JSON file. If you define an integration on the **Plugins** tab, add a reference to the integration name in the JSON file. If you define an integration directly in a JSON file described in Step 3, no additional configuration is required.

Github plugin integration example

Github plugin integration example

About this task

To configure the Github plugin integration, complete the following steps:

1. In the **Integration Name** field, enter a name for the integration, such as "Git tutorial."
   Such as GitHub tutorial. Later, we use this name to associate the integration to our value stream.
2. In the **Url** field, enter the URL for the GitHub project that you configured earlier .
   The URL is similar to this example, `https://github.com/myGitHubID/myGitHubProject`.
3. In the **Name** field, enter the name of the GitHub repository.
   Using the example URL from the preceding step, the name is `myGitHubProject`.
4. In the **Owner** field, enter the GitHub account that owns the repository.
   Using the example URL, the owner is `myGitHubID`.
5. In the **Token** field, enter your Github personal access token to authenticate with the Github repository.
   In GitHub, to generate a personal access token, go to **Settings**>**Developer settings**>**Personal access tokens** then click **Generate new token** and select the scope **repo** which allows to access repositories.
6. In the **API URL** field, enter the GitHub API URL.
   For enterprise Github account, use GitHub API URL `https://example.com/api/v3`.
   For public Github account, use GitHub API URL `https://api.github.com`.
7. Select the **Show Hidden Properties** check box.
   Result
   The hidden fields are displayed.
8. Enter the custom time for the plugin job in the **Plugin Job Timeout** field.

   Note: The default time is 1h, you can specify the maximum time a plugin job should be allowed to run before being automatically canceled in the following format:
   - 30m for 30 minutes
   - 1h for one hour
   - 1d for one day

9. **Optional:** Select the **Interval** time to set the frequency to run the schedule event.

   Note: If you do not want to change the interval time of the schedule event while creating the integration, the default sync intervals defined in the plugin are applied.

10. Click **Add**.

Results

The integration is listed on the **Integrations** page. If the integration is successful, the **Status** field displays `Online`. If the status is not `Online`, use the **View log** option to troubleshoot the integration. You can increase the logging level in the integration configuration if you need more troubleshooting information.

Managing integrations

You can manage the existing configured integrations.

About this task

After an integration is successfully configured, it periodically communicates with the external tool to synchronize the data flow.

The default communication interval when scheduled event integrations run can be configured by the `PLUGIN_INTEGRATION_RUN_INTERVAL` environment variable on the reporting-consumer service to a valid integer for the minutes between each run.

To manage an integration, perform the following steps:

1. On the Measure **Home** page, click **Settings > Integrations**.
   Result

   The **Integrations** page lists all configured integrations. All integrations are listed regardless of how they are created. You can use search functionality for searching the integration in the **Integrations** tab. The search query finds the matches in **Name** and **Type** fields. On the **Integrations** tab, you can sort the **Name**, **Status**, **Type**, and **Last sync** fields in ascending or descending order by clicking on the required table header field.
2. **Optional:** To modify an integration definition, click overflow menu ⋮ **> Edit**, and then modify the integration definition properties.
3. **Optional:** To view the integration log file, click overflow menu ⋮ **> View logs**.
   This action displays the integration's last execution log.
4. **Optional:** To upgrade or downgrade the plugin used by the integration, click overflow menu ⋮ **> Upgrade/ Downgrade**.
5. **Optional:** To sync with the external tool, click overflow menu ⋮ **> Resync**.
   This action attempts to communicate with the external tool immediately. You might do this when you want to run tests before the next communication interval. You can also select the time line from when you want to resync the data from the external tool.

   Note:

Not all integration types support this action. All scheduled event plugins such as Jira, GitHub, GitLab, Azure DevOps, and HCL DevOps Deploy are supported with the resync functionality.

6. **Optional:** To archive an integration, click overflow menu **> Archive/Delete**.

   Note: Select the **Delete the integration data** checkbox to delete the integration data.

7. **Optional:** To disable or enable an integration, click the **Disable/Enable** field of the integration.
   Result
   Communication with the external tool stops until re-enabled. You might do this if you want to stop data delivery during testing.
8. **Optional:** To disable or enable multiple integrations, select the required integrations and then **Select Actions** appears. Click the drop-down list of **Select Actions** and select **Disable** or **Enable**.

Defining an initial sync date for plugins

For scheduled events plugins, you can specify an initial sync date.

About this task

For scheduled events plugins, you can specify an initial sync date by performing the following steps:

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click **Plugins** tab.
3. Locate the required plugin and then click **Add integration**.
4. In the **Add Integration** window, click the **Show hidden properties** check box.
5. Click **Initial Sync Date** field to specify the date to pull data for initial run.
6. Fill the remaining required fields in the **Add Integration** window.
7. Click **Add**.

Note: If the plugin does not support the sync feature, the **Initial Sync Date** field is not displayed when you click the **Show hidden properties** check box in the **Add Integration** window.

Archiving integrations

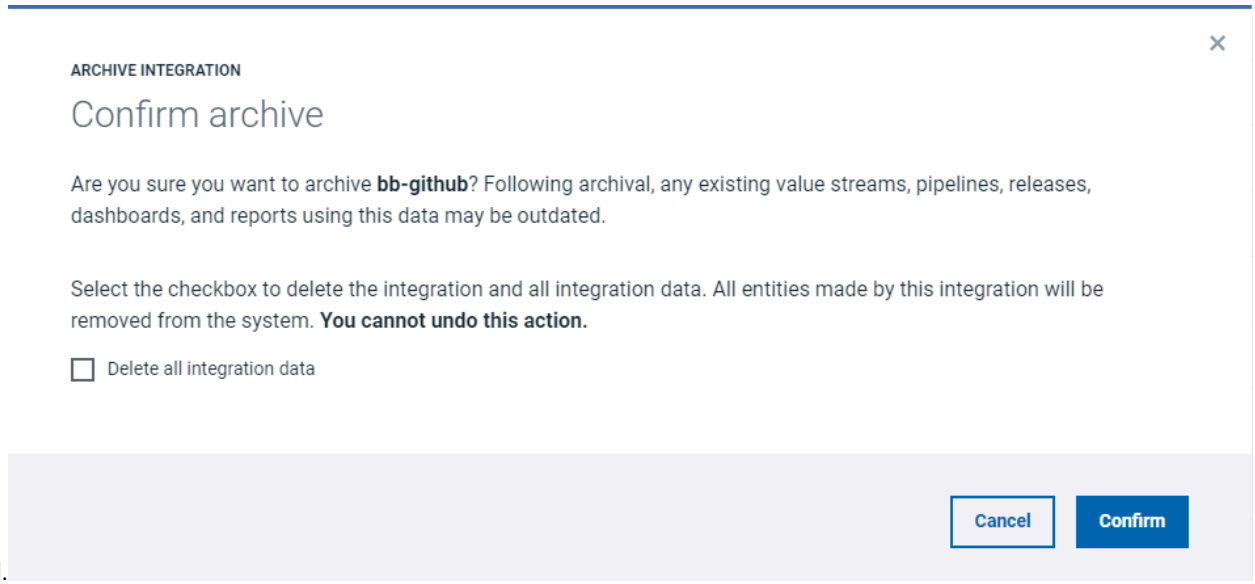
Archive the no longer needed integrations.

About this task

This topic describes the procedure for archiving integrations.

To archive integration, complete the following steps:

1. On the **Home** page, click **Settings > Integrations > Configured**.
   Result
2. Select **Archive/Delete** from the overflow menu (⋮) of the plugin integration which you want to archive.
3. The archive confirmation modal window is displayed.



4. Select **Delete all integration data** checkbox to delete all integration data and click **Confirm**.
   You can view all archived integrations by clicking **Show Archived**.

Note: You can restore the archived integrations by clicking the overflow menu and select the **Restore**.

Upgrading and downgrading integrations

You can upgrade or downgrade the integration versions based on the plugin's compatibility with Measure.

Before you begin

You must have installed the required plugin. See Creating plugin integrations on page 1876.

1. On the **Home** page, click **Settings > Integrations**.
   The **Plugin and Integrations** page is displayed.
2. Click **Integrations**.
3. Select the toggle button under the **Disable/Enable** field to disable the plugin integration for which you want to upgrade or downgrade the version.
   You can avoid a partial data synchronization issue by disabling the integration before the upgrade.
4. Select **Upgrade/Downgrade** from the overflow menu (⋮) .

Note: The plugin version must be compatible with the Velocity version. For more details about the compatibility matrix, see the specific plugin documentation.

5. Select the version from the drop-down list.
6. Click **Upgrade**.

Note: If you select a lower version of the plugin from the drop-down list, the integration downgrades to the lower version.

7. Select the toggle button under the **Disable/Enable** field to enable the plugin integration.

Results

You have upgraded or downgraded the integrations.

Creating plugins with the plugins SDK

You can create your own plugins and install them in Measure. After installing custom plugins, you can integrate them in your releases, reports, and value streams.

Measure provides a SDK that contains the tools, including local dependencies and wrappers, that you need to create your own plugins. The plugin SDK provides a sample plugin that you can use to get started, and JavaScript templates that you can customize for your own plugins.

To create a plugin, your development environment must include the following items:

- Node.js version 12 or later
- NPM version 6 or later
- Docker
- Measure version 2.0 or later

To get started with the plugin SDK, complete the following steps:

1. Download the plugin SDK. On the Measure **Plugins** tab, click **Download plugin sample**. After the download is finished, extract the compressed file.
2. Build the sample plugin. In the `ucv-ext-sdk-template` folder, run the following commands. Additional instructions are provided in the SDK `Readme.md` file.

```
 npm install
npm run dist-runnable
docker build . -t name:tag
```

The `name` value can be any value that you want. The `tag` value must represent the latest image version, or some other valid semantic version. For example, `docker build . -t testplugin:0.1`.

3. Upload the sample plugin. On the Velocity **Plugins** tab, click **Load plugin**, and then select the Docker image that contains your plugin. To upload the image from the previous step, you specify `testplugin:0.1`.
4. Create an integration by using the sample plugin. On the **Plugins** tab, click **Add integration**.

Plugins are categorized by data collection and communication methods. Generally, plugins are designed to use one of the following communication methods.

- **Webhook**. These plugins use webhooks to communicate with a defined Measure API endpoint. The webhook is used to trigger data collection events. Examples include the AppScan™, and SonarCube plugins.
- **Poller**. These plugins are based on an event defined by the plugin. Queries are performed to determine when to send and update data from the external service. Examples include the GitHub™, and Rally plugins.
- **Parser**. These plugins integrate functions to parse a specific file type and create a metric document that Measure can display with other value stream management and portfolio views. Examples include the JUnit, and OneTest plugins.

The plugin SDK provides JavaScript templates for each type. For example, the template `ucv-ext-sdk-templates/src/endpoints/SampleEdnpoint.js` defines the properties that you need to create an endpoint-type plugin.

Managing native integrations

Native integrations add external functionality to releases, deployment plans, and pipelines.

Native integrations are available for the following applications or services:

- IBM DevOps Deploy
- Jenkins™
- ServiceNow™
- Ansible™

Configuring DevOps Deploy integrations

Use DevOps Deploy applications in your deployments and pipelines.

Before you begin

- Obtain a token from your DevOps Deploy server. When you create the DevOps Deploy token, specify **admin** as the user.

  This task requires that you are an Measure administrator.
- You must install the DevOps Deploy plugin, see Github plugin integration example on page 1880.

Note:

You must not integrate the same instance of Deploy using UrbanCode Deploy (Legacy) and Deploy (ucv-ext-ucd) plug-in in Measure. Integrating the same Deploy instance using UrbanCode Deploy (Legacy) and Deploy plug-in in Velocity can result in data corruption.

About this task

After you configure an integration, create tasks to run DevOps Deploy applications during your deployments on page 1868. You can also add applications to your pipeline stages and run them that way.

1. Click **Settings > Integrations**.
   The **Plugin and Integrations** page is displayed.
2. Click **Installed Plugins**, and click **Add Integration** for **DevOps Deploy**.
3. In the **Add DevOps Deploy Integration** window, enter a name for the integration, and in the **URL** field, specify the URL. The format is `HTTPS://hostname:port`. For example, `HTTPS://sampleUCD:8443`.

   Note: If you are running DevOps Deploy locally, you can substitute `HTTPS://host.docker.internal:port` in place of `localhost`.

4. In the **Access Token** field, paste the token from DevOps Deploy.
5. Click **Add**.

   Note: The initial synchronization includes four months of data. Depending on the amount of data, the synchronization process can take some time.

What to do next

Synchronization occurs every fifteen minutes but you can synchronize any time by clicking **Actions > Resync** on the **Integrations** page. To prevent synchronization, select the toggle button under **Disable/Enable** field on the **Integrations** page.

Configuring Jenkins integrations

Manage Jenkins™ jobs with your deployments and pipelines.

Before you begin

Obtain the Measure plugin and install it in your Jenkins instance. On the Jenkins Plugin page search for `velocity`. Complete the integration by pasting the Measure credentials into Jenkins.

Note: Ensure the following requirements are met prior to installation of the plugin.

- You must have admin privileges in Jenkins.
- Upgrade the Jenkins plugin version to 2.1.0 in order to upgrade the Velocity version to 2.1.0 and later. Velocity version 2.0.6 will be the last version to support the previous Jenkins plugin version of 2.0.6 and earlier.

Note:

You must not integrate the same instance of Jenkins using Jenkins (Legacy) and Jenkins (ucv-ext-jenkins) plug-in in Measure. Integrating the same Jenkins instance using Jenkins (Legacy) and Jenkins plug-in in Velocity can result in data corruption.

About this task

This configuration makes Jenkins jobs available to pipelines and deployment plans. You can run Jenkins jobs with Measure and use the output in later tasks. After you configure an integration, add Jenkins tasks to deployment plans on page 1858 and pipeline stages.

1. Click **Settings > Integrations > Plugins**.
   A list of defined integrations with current information about each integration is displayed in a table.
2. Click **Add Integration** for **Jenkins**.
3. In the **Add Jenkins Integration** window, in the **Name** field, enter a name for the integration and click **Create**.
4. In the **Integration ID** field, click to copy the ID and then paste it into the corresponding field in your Jenkins instance, **Manage Jenkins > Configure System > DevOps Velocity**.
5. In the **Integration token** field, click to copy the token and then paste it into the corresponding field in Jenkins.
6. Click **Save**.
   Depending on the amount of data,the initial integration can take some time.
   Result
   Synchronization occurs every minute. To prevent synchronization, click the **Disable/Enable** field on the **Integrations** page.

What to do next

After the initial integration, you can add Jenkins tasks to deployment plans and select any of the integrated jobs. Similarly, you can add Jenkins jobs to pipeline stages.

Integrating with ServiceNow

Create and manage ServiceNow™ change requests with Measure. Update change requests with Measure tasks and then react to status changes. Control deployment progress with change request updates.

ServiceNow integrations are managed on the task level. You configure an integration when you add a ServiceNow task to a deployment plan on page 1859. After an integration is configured, other ServiceNow tasks can use the same instance.

Migrating integrations

Migrate the legacy plug-in integrations to external plug-in integrations.

You can migrate the following (Legacy) plug-in integration to external plug-in integrations:

Migrating integration of Jenkins (Legacy) to Jenkins

Migrating integration of Jenkins (Legacy) to Jenkins.

Before you begin

You must have completed the following tasks:

- Back up the database, see instructions in this page.
- Ensure that you have added the version parameter to all applicable jobs in Jenkins server. The Jenkins plugin search for jobs with version parameter as reference.

About this task

To migrate the integrations of Jenkins (Legacy) to Jenkins, complete the following steps:

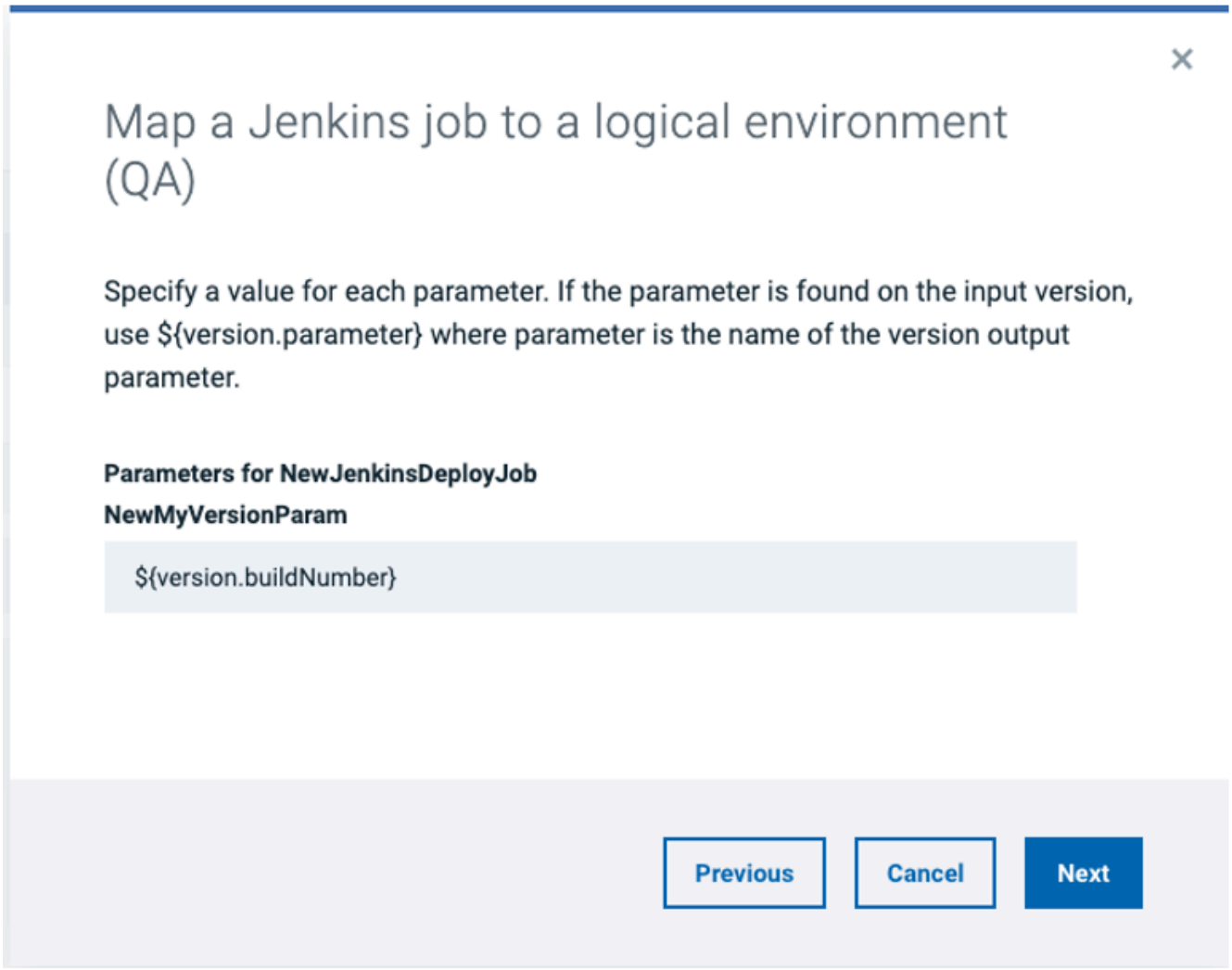
1. From the Measure page, click **Settings > Integrations > Configured**.
   A list of all configured integrations are displayed.
2. Click overflow menu for configured **Jenkins (Legacy)** plug-in and then click **Migrate**.
   The **MIGRATE JENKINS INTEGRATION** window is displayed. You can view the road-map of migration process. Before you begin the migration process, ensure that you have taken the backup of database.
3. On the **MIGRATE JENKINS INTEGRATION** window, click **Continue**.
   The **Add Jenkins Integration** window is displayed.
4. On the **Add Jenkins Integration** window, enter the value for all required fields to create new integration of **Jenkins** and then click **Add** to add the integration details.
   The migration process starts automatically and it reaches to **complete** step of migration process.
5. Click **Close** to close **Migrate Jenkins Integration window**.
   You have successfully migrated integration of **Jenkins (Legacy)** to **Jenkins**.

Results

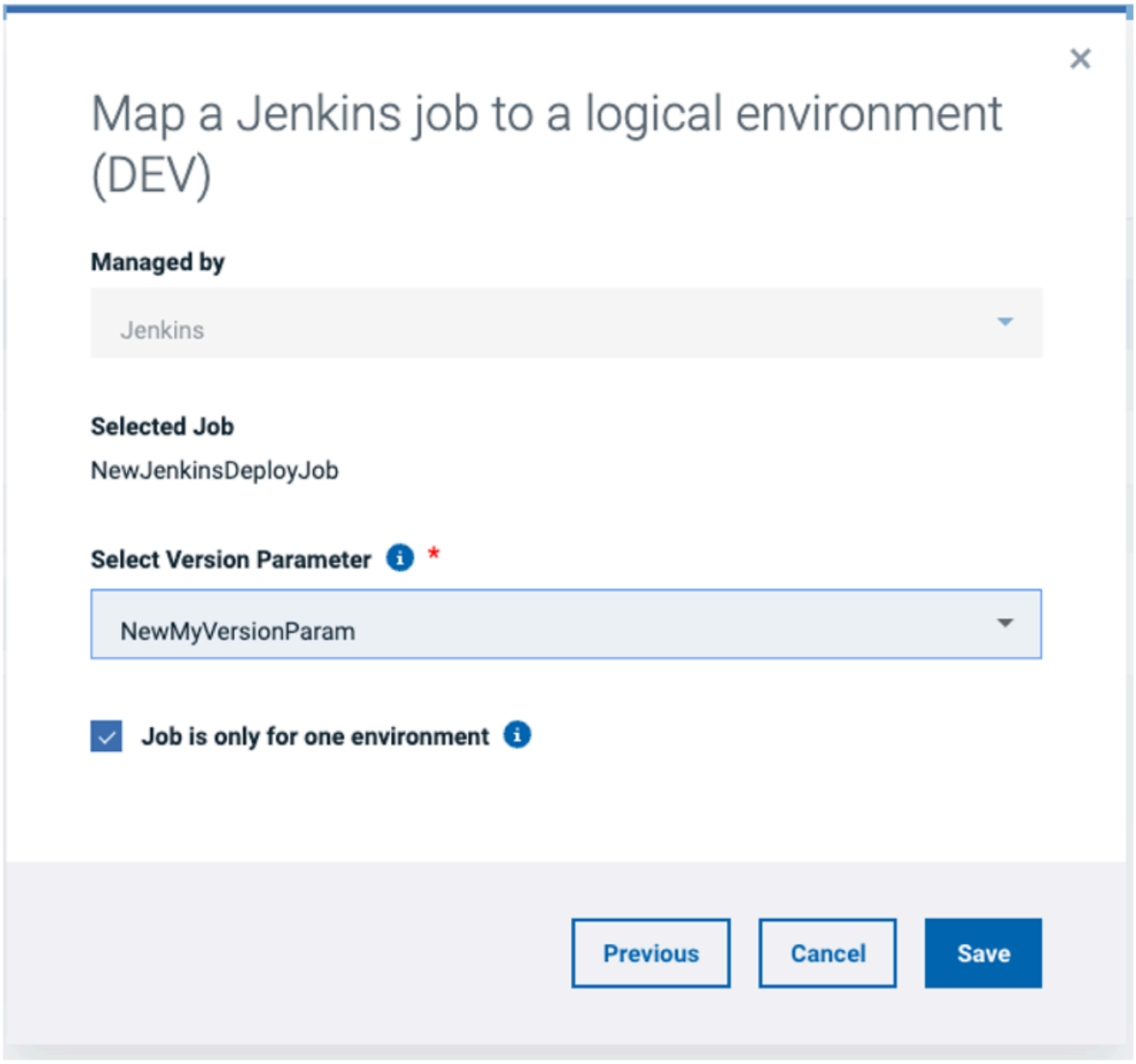
You can see the **Jenkins** integration data in Velocity.

What to do next

Go to pipeline of any value stream and add Jenkins application. While configuring the Jenkins job to logical environment ensure that you use `${version.buildNumber}` parameter.


Map a Jenkins job to a logical environment (QA)
Specify a value for each parameter. If the parameter is found on the input version, use ${version.parameter} where parameter is the name of the version output parameter.
Parameters for NewJenkinsDeployJob
NewMyVersionParam
${version.buildNumber}
Previous
Cancel
Next

Choose the required parameter in **Select Version Parameter**



field.

Migrating integration of UrbanCode Deploy (Legacy) to HCL DevOps Deploy (Deploy)

Migrating integration of UrbanCode Deploy (Legacy) to DevOps Deploy.

Before you begin

You must have completed the following tasks:

- Ensured that you have taken Back up the database, see instructions in this page.
- Ensured that there is no **Releases** with **UrbanCode Deploy (Legacy)** task is **In progress** or **Scheduled** for later.
- Ensured that there is no **UrbanCode Deploy (Legacy)** deployment is **In progress** or **Scheduled** for later.

About this task

To migrate the integrations of UrbanCode Deploy (Legacy) to DevOps Deploy, complete the following steps:

1. From the Measure page, click **Settings > Integrations > Configured**.
   A list of all configured integrations are displayed.
2. Click overflow menu for configured **UrbanCode Deploy (Legacy)** plug-in and then click **Migrate**.
   The **MIGRATE URBANCODE DEPLOY(HCL LAUNCH) INTEGRATION** window is displayed. You can view the stages of the migration process.
3. On the **MIGRATE URBANCODE DEPLOY(HCL LAUNCH) INTEGRATION** window, click **Continue**.
4. The **URBANCODE DEPLOY (Legacy)** integration is migrated to DevOps Deploy integration. The integration name is changed by appending **--migrated-EXT**.
   The migration process automatically installs latest available version of the DevOps Deploy plugin. If any version of DevOps Deploy plugin is already installed, the migration process uses that version of the plugin for migrating the integration.

Results

All **URBANCODE DEPLOY (Legacy)** task type changes to DevOps Deploy task in templates.

What to do next

You can schedule the deployment for DevOps Deploy applications by using the migrated DevOps Deploy integrations.

Uploading a build to a pipeline

builds/apiDriven

```
curl -k -X POST 'https://localhost/api/v1/builds/apiDriven' \
    -H 'Authorizat1on: UserAccessKey <User Access Key>' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":"<Version Name>",
        "application":{"externalId":"efecb73f-dclc-4d3d-b3bb-
        38339445e892"},
        "url":"<Build URL>",
        "revision":"<Commit SHA>"
    }'
```

Table 674. builds/apiDriven parameters

Parameter	Type	Required	Description
UserAccesKey	string	false	User access key
version	string	true	JSON object with either 'externalId' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})

Table 674. builds/apiDriven parameters (continued)

Parameter	Type	Required	Description
application	object	false	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
url	URL	false	URL link to build in external tool
revision	string	true	SHA of the commit linked to the build

Example

Example

```
curl -k -X POST 'https://localhost/api/v1/builds/apiDriven' \
    -H 'Authorizat1on: UserAccessKey 3b3ec6cc-1e37-4353-bc3a-46b493eed4e6' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":"1.0",
        "application":{"externalId":"b900ff5b-1fff-44ab-bcc6-9d39d5937026"},
        "url":"https://test.com",
        "revision":"12345678"
    }'
```

Uploading deployment status

deployments/apiDriven

```
curl -k -X POST 'https://localhost/api/v1/deployments/apiDriven' \
    -H 'Authorizat1on: UserAccessKey <User Access Key>' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":{"name":"<Version Name>",
        "application":{"externalId":"efecb73f-dclc-4d3d-b3bb-
        38339445e892"},
        "url":"<Deployment URL>",
        "environment":{"name":"DEV"}
    }'
```

Table 675. deployments/apiDriven parameters

Parameter	Type	Required	Description
UserAccesKey	string	false	User access key

Table 675. deployments/apiDriven parameters (continued)

Parameter	Type	Required	Description
version	object	true	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
application	object	true	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
url	URL	false	URL link to build in external tool
environment	object	true	JSON object with either 'externalId' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})

Example

Example

```
curl -k -X POST 'https://localhost/api/v1/deployments/apiDriven' \
    -H 'Authorization: UserAccessKey 324adfe9-cc97-43fa-81c6-e248ceeb726a' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":{"name":"1.0",
        "application":{"externalId":"f2262e67-6516-4597-92a6-cfe537abdd00"},
        "url":"http://example.com",
        "environment":{"name":"DEV"}
    }'
```

Getting started with GraphQL in Measure

You can use GraphQL Playground in explore and test APIs interactively before using them in real scenarios. This GraphQL playground provides a convenient environment to use queries and mutations, that helps you understand the available data and how to structure requests effectively.

GraphQL is a flexible and efficient query language for APIs that allows you to request exactly the data you need. Unlike traditional REST APIs that return fixed data structures, GraphQL helps you define the shape of the response, fetch related data in a single request, and interact with data using queries (to read data) and mutations (to create or update data).

With GraphQL in Measure, you can use queries to retrieve value stream data, such as work items, deployments, commits, and other DevOps artifacts, and use mutations to create or manage resources.

Setting up GraphQL with API endpoints

Before you begin, you must create a user access key. See Managing User access keys (UAKs) on page 1820.

Follow the steps to set up and authenticate the GraphQL Integrated Development Environment (IDE) in DevOps Velocity:

1. Navigate to the Velocity instance GraphQL interface using `https://my_velocity:port/graphiql`.
2. Click the Set Headers icon.
3. Enter **Authorization** in the **Header key** field.
4. Enter the user access key in the **Value** field.

After the authentication is complete, you can start using different GraphQL APIs to validate responses and refine your queries. The GraphQL Playground also provides built-in documentation for each query and mutation, allows you to easily explore available fields, input parameters, and response structures.

After testing, you can use these APIs in your applications or leverage them to create and manage resources in Measure for bulk operations or automation scenarios without using the UI.

Mutations

Mutations are used to create new data, update existing data, and delete existing data.

Here are a few examples of mutations that you can use in Velocity.

- Create a Value Stream

  A Value Stream represents the flow of work items through a pipeline. It helps you visualize how work items move from one stage to another across various tools in the pipeline. To create a new value stream, you can use the following GraphQL mutation:

  ```
  mutation {
    createValueStream(
      name: "string"
      description: "string"
      tenantId: "_____"
      team: { id: "string", name: "string" }
      workflow: "_____"
    )
  }
  ```

- Add an environment to a pipeline

  A pipeline defines the series of steps required for deploying software, while stages represent the different phases of the pipeline (for example, build, deploy, test). To add a new stage to a pipeline, you can use the following GraphQL mutation:

  ```
  mutation {
    addPipelineStages(
      pipelineId: "Pipeline_ID",
      stages: [{ name: "<Stage_name>", type: "<Stage_type>" }]
    ) {
      _id
      name
      type
      description
      planId
      triggers {
  ```

```
      _id
      name
    }
    gates {
      _id
      name
    }
    cf_space_id
    promoteTo
    latestAutopromotionStatus
    deploymentModes
    defaultDeploymentMode
    groupsAndUsers
    solutionVersionId
  }
}
```

- Add an application to a pipeline

  To add an application to a pipeline, you can use the following mutation:

```
mutation {
  addApplications(
    pipelineId: "_____"
    applications: [{ name: "string" }]
    processes: [{ name: "string" }]
    type: "string"
  ) {
    _id
    name
    tenant_id
    description
    stages {
      _id
      name
      type
      description
      planId
      triggers
      gates
      cf_space_id
      promoteTo
      latestAutopromotionStatus
      deploymentModes
      defaultDeploymentMode
      groupsAndUsers
      solutionVersionId
      solutionVersion
    }
    applications {
      _id
      name
      type
      description
      processes
      source
      inputJobId
      stages
      subApplications
```

```
      children
      level
      external_id
      selectedChildren
      versionRegEx
    }
    synchronize
    team {
      id
      name
      tenantId
    }
    created
    stats {
      leadTime
    }
  }
}
```

Queries

You can access detailed information from Measure by using GraphQL queries.

The following are some example queries that demonstrate how to retrieve various types of data.

- Get application data by using the application ID

  You can retrieve detailed information about a specific application by using its ID and the following query:

```
{
  applicationById(id: "<Application_ID>", tenantId: "<Tenant_ID>", loadChildren: true) {
    _id
    id
    external_id
    tenant_id
    integration_id
    name
    alphaNumericName
    active
    createdAdHoc
    json_data
    type
    version
    level
    childrenIds
    children {
      _id
      id
      external_id
      tenant_id
      integration_id
      name
      alphaNumericName
      active
      createdAdHoc
      json_data
      type
```

```
      version
      level
    }
  }
}
```

- Retrieve the application environment details by tenantID

  To retrieve all unique application environment details for a given tenantID, you can use the following query::

```
{
  applicationEnvironmentsByTenantId(tenantId: "TenantID") {
    _id
    id
    external_id
    tenant_id
    name
    app_id
    external_app_id
    active
    json_data
    type
    version
    inventory {
      external_version_id
      components
    }
  }
}
```

- Retrieve application environment name by tenantID

  To retrieve all unique application environment names for a given tenantID, you can use the following query:

```
{
  uniqueApplicationEnvironmentNamesByTenantId(tenantId: "_____") {
    name
  }
}
```

How to

The topics in this section describe how to work with Measure features. Tutorials and workbooks are available for users of all experience levels.

Creating reports

Use the **Home** page to create one of the following reports.

About this task

- Custom
- Delivery Transformation
- Recent Activity

- Team Performance
- Audit

1. From the **Home** page, click **Insights > Reports**.
2. Click to **Add report**.
3. On the **New report** page, in the **Type of report** field, select the type of report from the list.
4. Enter the name of the report in the **Report name** field.
5. Enter a report description in the **Description** field.
6. Select an icon for the report in the **Icon** field.
7. Click **Add**.
   After the report is created you can immediately run, edit, and share the report with other users.

Sharing reports

You can share reports with others in your organization and who are authenticated to Measure. By default, you see only the reports that you create.

About this task

When you share a report, the users you are sharing the report with are able to access the report from their Home page. Depending on the access level that you provide, they can modify the report or only view report data.

Note: The data users can view is restricted to data from the teams of which they are a member.

You can remove access at any time by clicking the delete icon next to the email address on the **Share Report** page.

1. From the **Insights > Reports** page, select the report that you want to share.
2. Click

to copy the URL link for sharing the report.
3. Share the URL of the report to required users.

   Note: The user must be authenticated for accessing the reports.

Glossary

This glossary provides terms and definitions for the Measure software and products.

The following cross-references are used in this glossary:

- *See* refers you from a non-preferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

A

activity

Activities include releases, events, and deployment plans that you develop, start, and complete with the product.

agent

A program that runs on a remote computer and communicates with the Measure server. Agents run processes on deployment target systems.

agent relay

A communication proxy for agents that are located behind a firewall or in another network location.

API

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

Application

One or more computer programs or software components that provide a function in direct support of a specific business process or processes. See also application server.

application environment

A user-defined collection of resources that hosts an application. These application environments refer to environments that are created to be in the product.

application process

A process that is associated with an application. Unlike a component or generic process, an application process is created from application-level steps. Typically, an application process invokes component processes, and orchestrates multi-component deployments. See also component process on page 1900, generic process on page 1901, process on page 1903.

artifact

A deployable item such as a file, image, database, configuration material, or anything else that is associated with a software project. By default, artifacts are stored in CodeStation repository.

authentication realm

An authentication realm manages users and determines user identity. It retrieves information from sources such as LDAP servers and Keystone identity services.

auto task

A task that starts automatically they are eligible to run. Examples of auto task types are HCL DevOps Deploy, Continuous Delivery pipeline, email, and Slack.

B

blackout

A setting that prevents deployments and snapshots from being scheduled. Blackouts solely affect the environment and application that they are applied to.

C

change

An artifact that represents a generalized notion of a development work product, such as a task, defect report, or enhancement request. See also initiative on page 1901.

component

A representation of deployable items and the user-defined processes that operate on them, usually by deploying them.

component process

A process defined for the deployment of components.

component version

See version on page 1904.

D

dashboard

A displayed collection of user-defined data filters and metric charts. Dashboards are used to analyze project data for their quality and efficiency.

dependent task

A task depends on another task to finish before it starts. The task that a dependent task waits for is called a prerequisite task. Dependent tasks cannot start until all its prerequisite tasks are resolved. Dependent tasks are used to ensure that tasks run in the specified order.

deployment

The activities used to deliver a software project to a deployment target. Typically, you run deployments for each stage of your release lifecycle, ending with the production stage when the software becomes generally available. The activities that are performed during a deployment, called tasks, are defined in deployment plans. You start a deployment by starting one of the plan's eligible tasks. You complete a deployment by resolving all the tasks in the deployment plan.

dots

A value stream unit of work, such as a pull request, build, or deployment. Units of work can be combined, such as commits and pull requests, which significantly enhances the data quality of value streams.

duration

The time a task takes to run. Duration is measured from the time a task starts until it is resolved. When you create some task types, you can estimate its expected duration. Duration is reported in minutes.

E

environment

A collection of resources that identify the components that can be deployed by the parent application and the agents that do the work.

events

Release-related activities that are applied to releases and tracked with a calendar. You can use events to organize your releases and other time-dependent activities, such as holidays and blackouts.

execution pattern

The order in which tasks in a deployment plan become eligible to run. By default, a plan's execution pattern is sequential. Sequential tasks run in order beginning with the first task in the plan. You can combine tasks into groups and assign the parallel execution pattern to the group. Parallel tasks can run in any order and run simultaneously.

G

gate

A requirement that must be met before an application can move on to the next stage.

generic process

Processes that are designed to run outside typical component or application processing. See also application process on page 1899, component process on page 1900, process on page 1903.

group

A logical container that serves as a mechanism to grant permissions to multiple users.

I

initative

An action to take for the change occurred.

insights

A set of related features such as metric data dashboards, reports, and the application portfolio.

integration

Regular communication between Measure and external products and services. Communication with integrated products can be bidirectional.

L

lifecycle

The phases in a release. A lifecycle is a template for the stages of work in a release.

lock

A means of preventing changes being made to a release.

logical environment

A collection of physical environments on page 1902.

M

manual task

A release related task that users must manually complete. See also auto task on page 1899.

metric data

Tools, such as charts and graphs that are applied to project data to assess project quality and progress.

metric definition

A JSON object that defines expected value types and chart default settings.

milestone

A significant event in a project or process that is used to determine progress toward goals.

P

particles

See dots on page 1900.

phase

An organizational segment within a value stream containing the stages, such as "Planning," or "Development,", that define process flow within that group.

physical environment

An actual deployable location, server, hardware, and so on.

pipeline

A sequence of stages that retrieve input and run applications and jobs, such as builds, tests, and deployments.

plugin

A separately installable software module that adds function to an existing program, application, or interface.

portfolio

A collection of application information in one location where you can analyze the data with metric charts and graphs.

process

In Measure, processes define automation activities, such as deploying components.

property

A variable that specifies a value that is provided during process deployment and is associated with a specific Measure object. An individual property's value can be accessed only by certain types of related objects, such as environments, processes, and components.

Q

quality

A user-defined attribute that serves as a gate for applications to enter phases.

R

release

The delivery of multiple applications through one or more phases of a lifecycle.

resource

A user-defined construct that is based on the architectural model of Measure. A resource represents a deployment target.

role

A job function that identifies the tasks that a user can perform and the resources to which a user has access. A user can be assigned one or more roles.

S

segment

A period of time in a deployment plan. Deployment plans can group tasks into segments to specify when tasks are run relative to each other.

snapshot

A collection of specific versions of components. Typically, a snapshot represents a set of component versions that are known to work together.

stage

In the value stream, a subset of a phase that contains data from integrated tools.

swim lane

A representation of value stream data in tabular form where the columns represent value stream stages, and the rows represent users.

T

tag

A label that you can apply to tasks or releases. When applied to tasks, tags can determine task applicability for a given deployment. When applied to releases, tags can be used to organize and filter releases.

task

Represents a business-meaningful activity that has starting and ending points and a measurable duration. Durations are used to estimate deployment times. You add tasks to deployment plans. When you run a deployment, you complete the tasks in the plan

task group

A combination of two or more tasks that are associated into a group. When you form a group, you define the group's execution pattern as sequential or parallel.

team

A team is a collection of users and groups.

token

A particular message or bit pattern that signifies permission or temporary control to transmit over a network.

U

user

A representation of an account on the server. Users can be members of teams and groups. User can be created in Velocity Measure server or import users from an external authentication realm, including an LDAP, Active Directory, or SSO provider.

V

value stream

A representation of the objects. The value stream provides a strategic window into your project workflow.

version

A representation of an Deploy application snapshot.

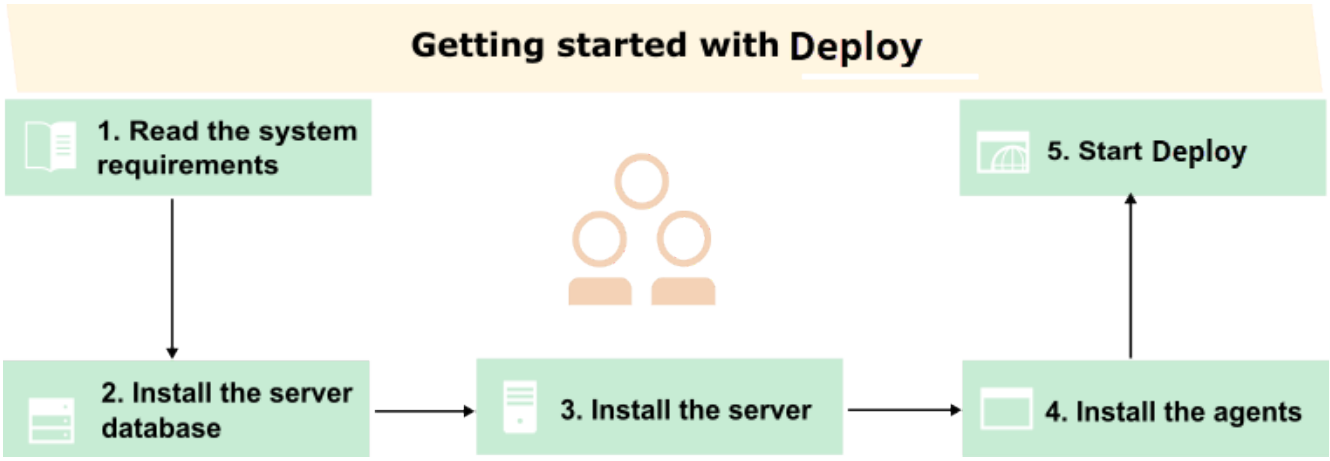
Deploy

Deploy standardizes and simplifies the rollout of software components across environments within the DevOps Loop. Its release automation tools improve deployment speed and reliability, providing visibility into multi-tiered deployments and orchestrating complex processes across environments and approval gates.

Getting Started

Quickly become productive with HCL DevOps Deploy (Deploy).

The following diagram outlines the steps that are required to get started with Deploy.



Read the system requirements

For supported platforms and requirements, see the System Requirements topic.

Install and start the server

To download the product, visit the HCLSoftware download portal website.

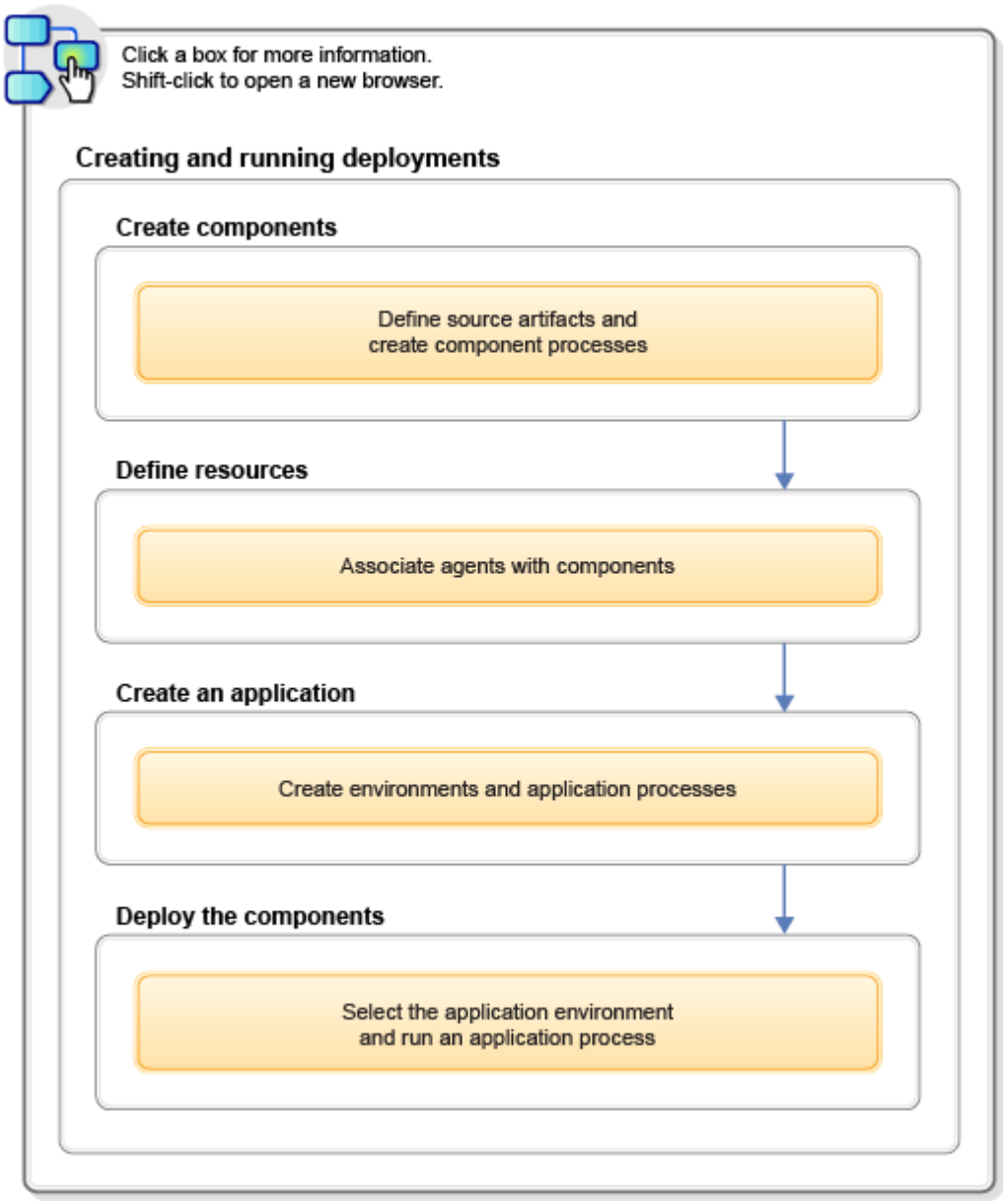
See the Installing topic for installation details.

For information on installing licenses, see Applying HCL licenses to agents on page 2115

Using Deploy

The following diagram outlines the minimum steps that are required to create and run a deployment. Each box describes a core activity and, taken together, they illustrate the product's primary function. The fastest way to become productive is to work through these steps and understand what each does and how each interrelates with the others.

From the **Welcome** page, you can work through a wizard that shows you the basic steps to deploying artifacts with Deploy. Click the **Welcome** tab, and then click **Let's creat an application**. The wizard guides you step-by-step and provides brief explanations of the entire process for creating a deployment. If you do not see the **Welcome** tab, click the help menu icon that is located in the upper-right corner of the window. Then, click **Welcome**. See Setting up the wizard for new users on page 2119 for configuration information.



1. Click this area for information about creating components. on page 2252
2. Click this area for information about creating resources. on page 2313
3. Click this area for information about creating applications. on page 2345
4. Click this area for information about running deployments. on page 2525

Overview of DevOps Deploy

Use HCL DevOps Deploy (Deploy) to standardize and simplify the process of deploying software components to each environment in your development cycle.

Deploy helps you meet the challenge of deploying software throughout your enterprise by providing tools that improve both deployment speed and reliability. The release automation tools in Deploy provide complete visibility into n-tiered

deployments. Use the tools to model processes that orchestrate complex deployments across every environment and approval gate.

By using plug-ins in Deploy, you integrate the server with your existing middleware and organize code from your existing code repositories into components. You design processes to deploy your code by using steps that run each required task. Then, you organize your components into applications and create environments for each stage of your release process. Finally, you create an application process to set the deployment order of your components and deploy your application to an environment.

Systems and topology overview

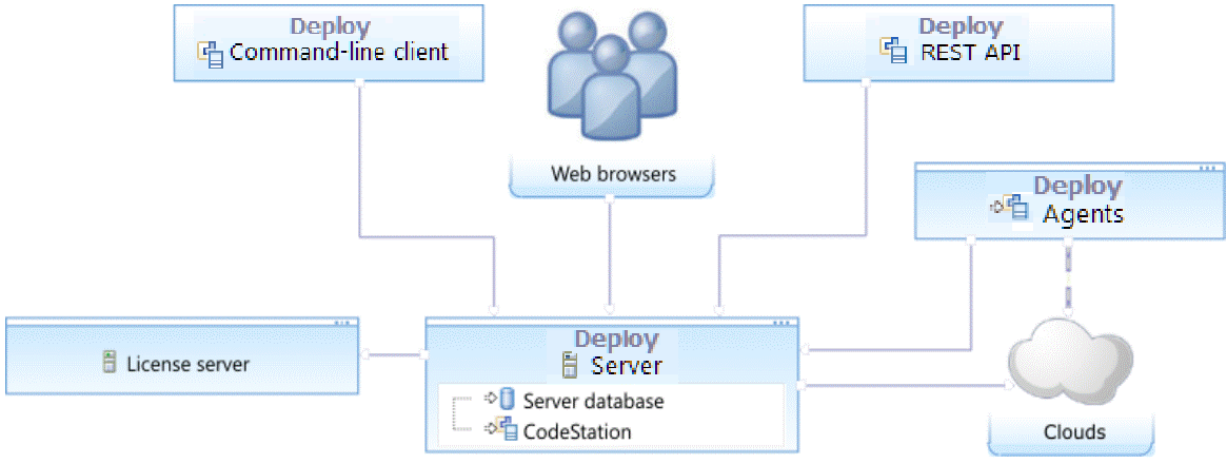
HCL DevOps Deploy (Deploy) includes several systems, including a server and one or more agents. You can configure multiple topologies, including ones that use high availability, disaster recovery, and so on, to meet your needs.

The following system topologies, including diagrams, are shown. For explanation of the system components, see the Description of systems on page 1911.

- Core topology on page 1907
- Multi-region topology on page 1908
- High-availability clustered topologies on page 1909
- Disaster recovery topologies on page 1909
- Default ports on page 1910
- Description of systems on page 1911

Core topology

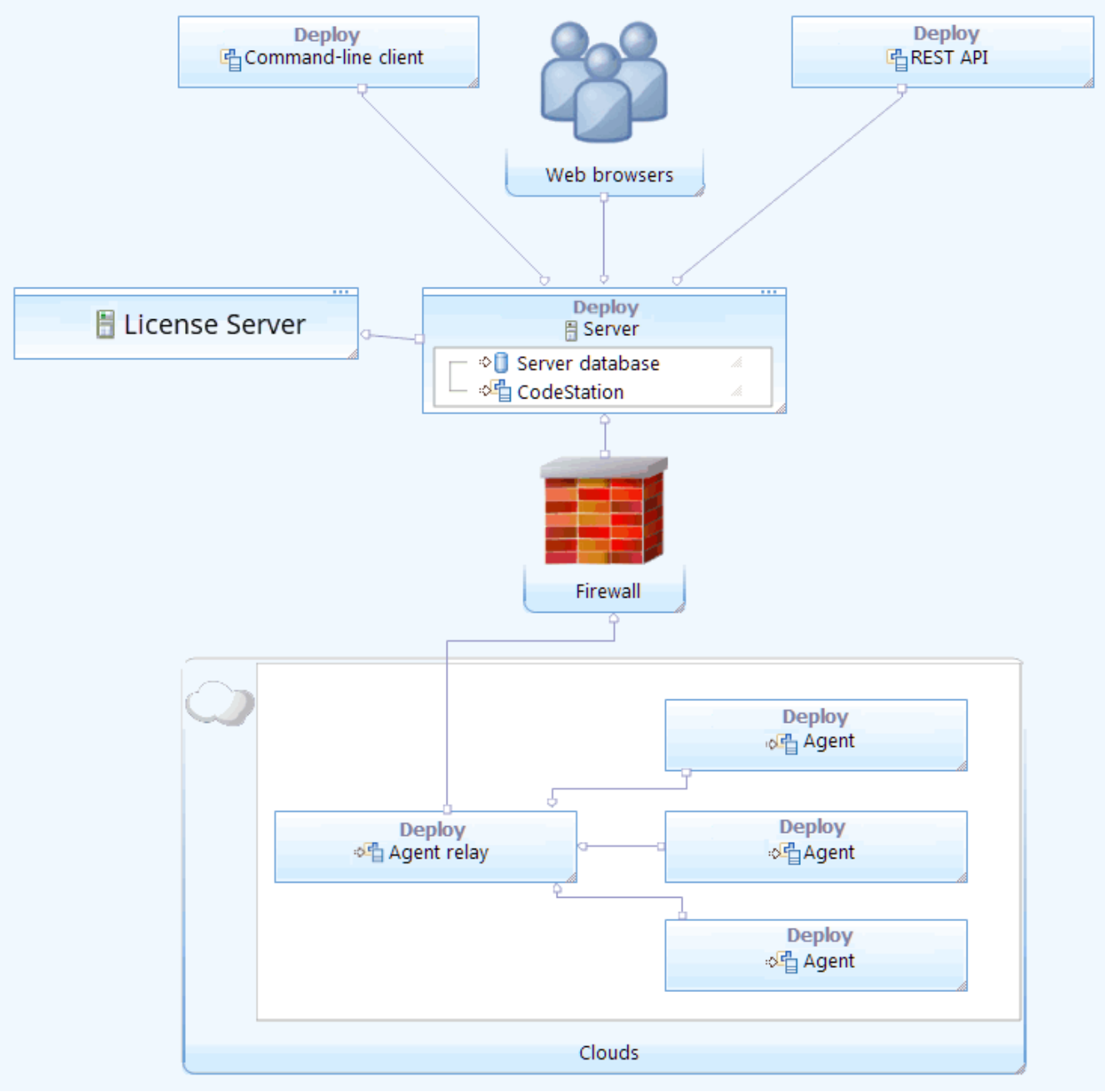
The core installation of Deploy includes a server, agents, and a license server. Clients access the server through web browsers, the REST API, or the command-line client. Agents can be installed on cloud environments, virtual machines (VMs), containers, or physical systems; the agents are shown on cloud systems in the following diagrams, but they can be installed on many different systems.



To install the core components, see Installing DevOps Deploy.

Multi-region topology

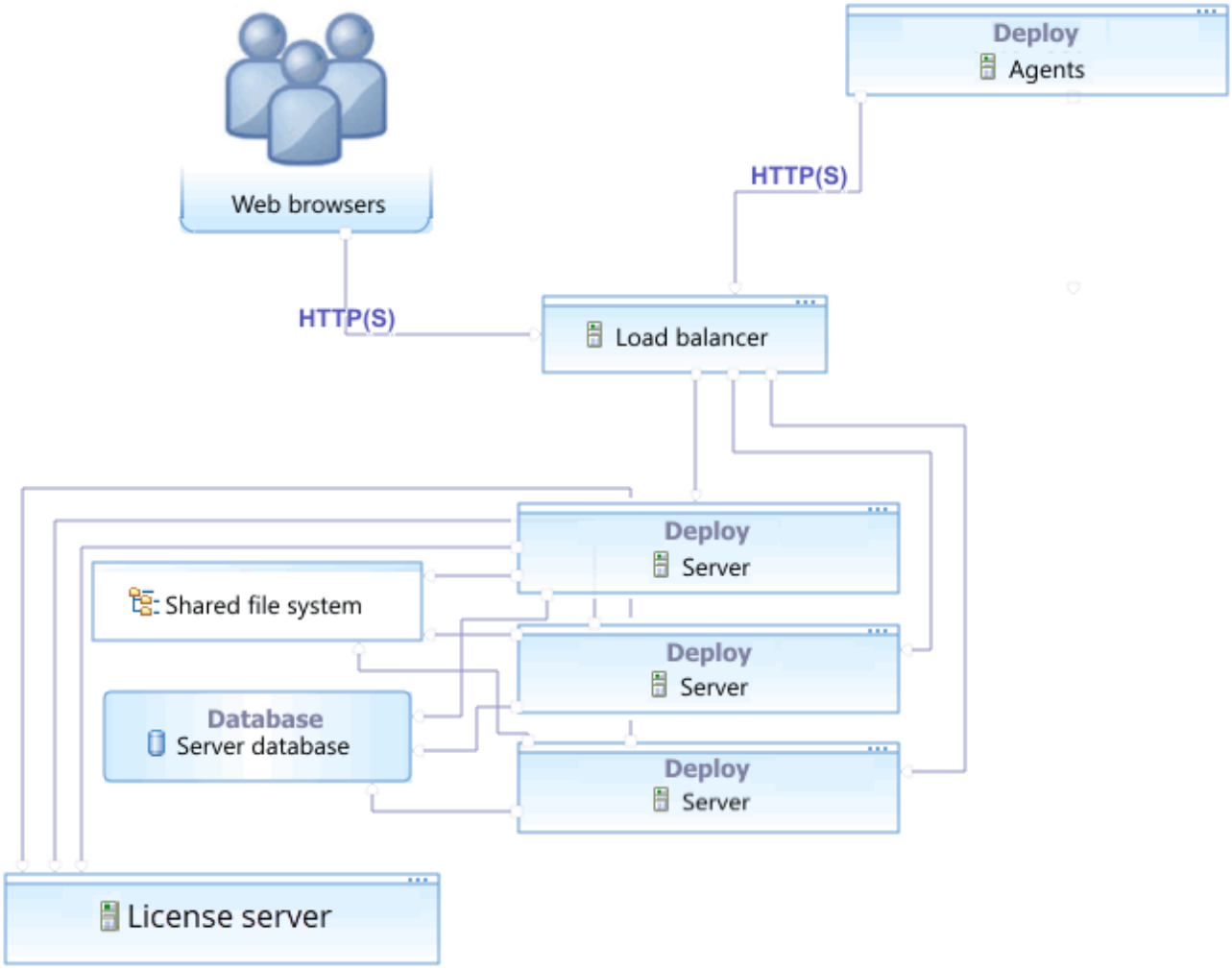
If your environment has multiple security zones that are divided by firewalls, you can use agent relays to connect agents to the server through the firewalls. For example, if your Deploy server is within a firewall, but your target environments are outside the firewall, the agents on those target environments cannot connect directly to the server. In this case, you install an agent relay outside the firewall to allow the agents to connect to the server through the firewall, as shown in the following diagram.



To install and configure an agent relay, see Installing agent relays.

High-availability clustered topologies

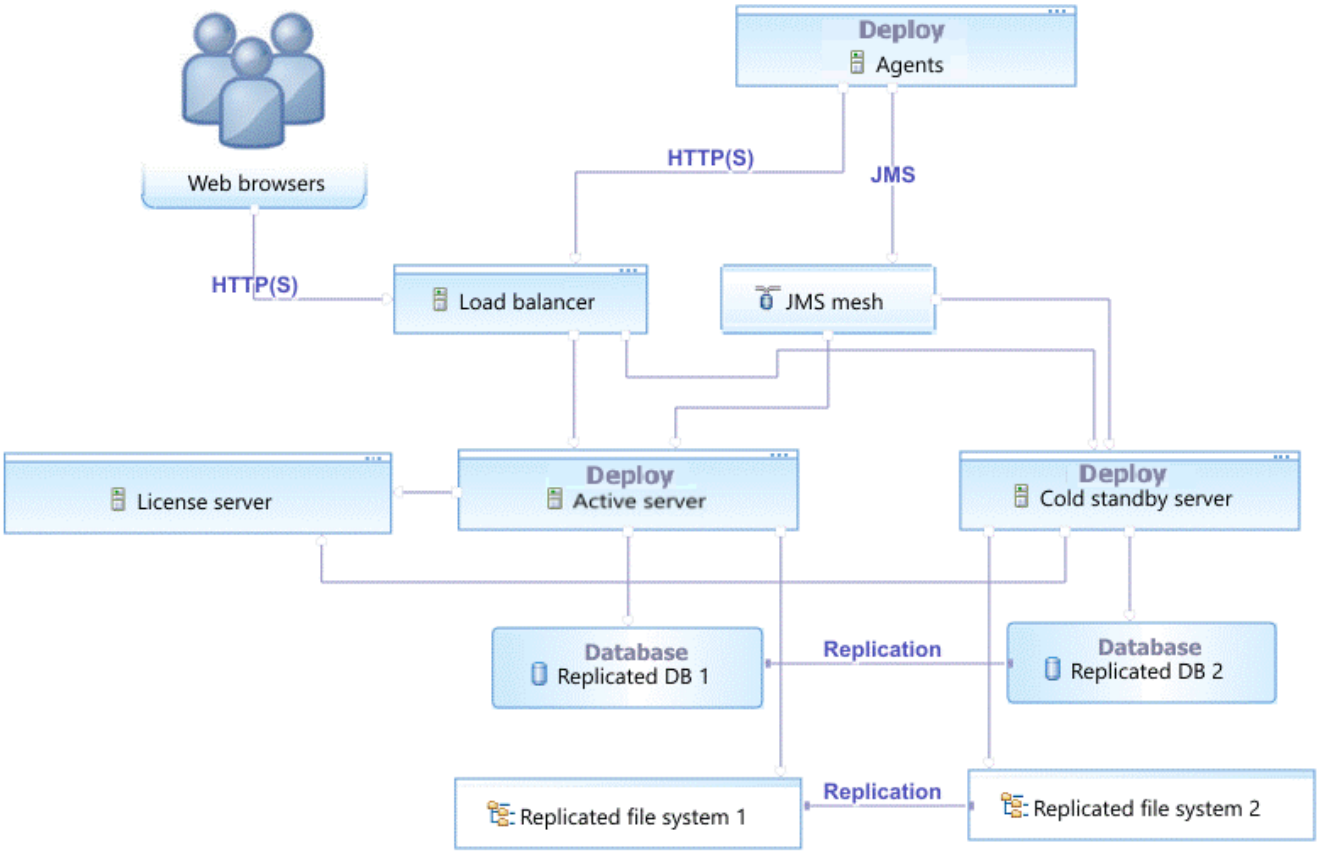
High-availability topologies use multiple servers. These servers can all be running at the same time to share the load (as in a clustered topology), or they can be waiting for another server to fail (as in a cold standby topology). The following diagram shows a clustered topology in which a load balancer distributes connections to three servers. Users connect directly to the load balancer, which sends them to an active server. Agents connect via HTTP and HTTPS to the load balancer. The servers store their files on a shared database and file system.



To configure a cluster of servers, see Setting up clusters of servers on page 2063.

Disaster recovery topologies

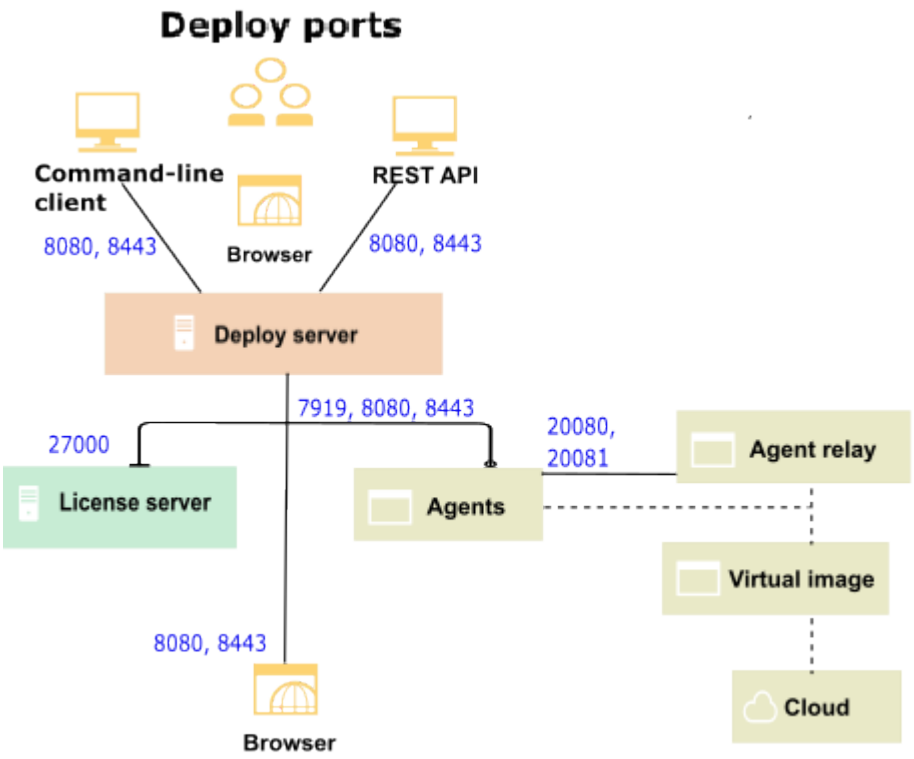
One way to prepare for disaster recovery is to have a cold standby system, including a stopped server and a replicated copy of the database and file system. The following diagram shows a simple topology with a cold standby server and related resources on standby.



To configure a cold standby system for the server, see Adding cold standby servers on page 2068.

Default ports

The following diagram shows the default port numbers that Deploy uses for communication. Most of these ports can change depending on your choices at installation time. The following diagram is only a summary of the defaults.



For more information about ports, see System requirements and performance considerations.

Description of systems

Deploy server

The Deploy server stores components, processes, and other elements with which you model automated application deployments. You run automated deployments from the server.

Web browsers

Web browsers are the main way that users interact with the server. The Deploy browser-based GUI is a Rich Internet Application (RIA) that maintains much of its functionality in the browser. Clients interact with RESTful (representational state transfer) services on the server as needed.

Command-line client

The command-line client provides access to the server through the command line. It can automate functions on the server, such as creating components and applications, and provides most of the features that are found in the browser-based GUI. The command-line client is also built on top of RESTful services. See Command-line client (CLI) reference on page 2591.

REST API

The REST API provides access to the server through HTTP. Like the command-line client, it can automate functions on the server, such as creating components and applications.

Agents

The agents run processes on target systems. Agents can run on physical computers, virtual systems, or cloud systems. See Agents on page 2336.

Agent relays

An agent relay is a communication proxy for agents that are located behind a firewall or in another network location. See Agent relays on page 2341.

License server

The license server provides licenses to the server.

Clouds

The clouds host virtual resources. For more information, see Connecting to clouds through Deploy.

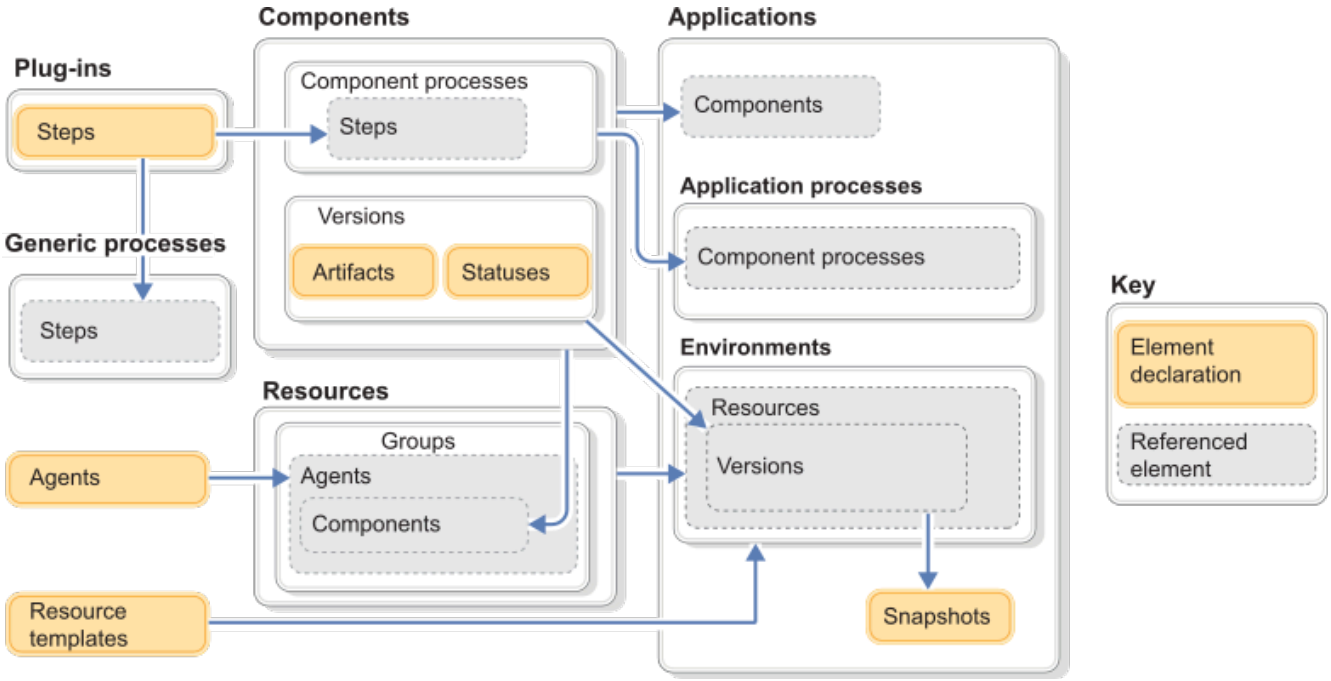
CodeStation

CodeStation stores component versions and artifacts. It is part of the Deploy server.

Elements overview

The HCL DevOps Deploy (Deploy) elements that you use to model software deployments include applications, environments, and components.

This graphic shows the relationships between different elements on the server, such as applications, environments, and components. Each element is described in the following text.



Components

Software components are central to Deploy. Components represent deploy-able items, also called artifacts, which can be files, images, databases, configuration materials, or anything else that is associated with a software project. Each time that you import new files that are associated with a component, a new component version is created. The server keeps copies of these component version artifacts, and you specify which component version to use each time you deploy. You can also apply statuses, such as `Selenium tests passed` to component versions and use the statuses in processes to control deployment behavior.

You create component processes, processes that usually contain all the steps that are necessary for the server to deploy the component to an environment. These steps describe all actions, such as starting a Tomcat server or updating files in a database. Many process steps are available by default, and you can obtain more steps from plug-ins.

Plug-ins

Automation plug-ins are the most common type of plug-in, and they provide more steps for component processes. For information about specific plug-ins, see the HCL Plug-ins site.

Resources

Resources are represented in a tree throughout Deploy to represent the relationships between components, the agents that deploy the components, and the target deployment environments. You create groups to represent target servers and environments, attach the agents that deploy to the environments, and attach components to the agents that run their processes.

The most common resource is an agent, a software program that facilitates communication between the component's host and the server. The agent runs the processes that deploy components.

To replicate a section of the resource tree, create a resource template. Resource templates describe a small section of the resource tree and can be applied to environments through a blueprint.

Applications

An application is a logical group of components that are deployed together. Applications have processes, but their steps usually start processes or modify resources. An application contains environments, which describe all the deployment locations that are needed in your deployment process, and you run application processes to deploy components to environments.

Deploy provides you with tools to ensure consistent deployments, like snapshots. Snapshots record the component versions that are deployed to an environment, and you can use a snapshot to select the component versions that are applied during a deployment.

Generic processes

You create a generic process to run any automated task that is not tied to a specific component or application. You can use any process step that is available for component processes when you create generic processes. You can create generic processes to set up new target servers or to perform audit tasks.

Learning

Tutorials and instructions for all experience levels.

The Tutorial section contains tutorials for users of all experience levels. If you are new to Deploy, the HelloWorld tutorial on page 1915 introduces the most-used product features in an easy-to-follow way. You do not need additional resources to complete the tutorial.

The How to section provides short, directed instructions for a variety of product features, such as how to configure web hooks.

Additional information about the product including descriptions of key features, tool integrations, and to schedule a demo, see the HCL DevOps Deploy product page. You can **Join Webinar** to view recorded videos about the product.

Visit the blogs page to learn more about the product features. Use the **Filter by** to see only Deploy topics.

Tutorials

Tutorials are available for every experience level.

This section contains tutorials for users of all experience levels. If you are new to Deploy, the HelloWorld tutorial on page 1915 introduces the most-used product features in an easy-to-follow way. You do not need additional resources to complete the tutorial.

After HelloWorld, the next step is the JPetStore on page 1934 tutorial that shows you how to deploy a data-driven web application. In addition to providing more in-depth topics, the JPetStore tutorial is also used by other tutorials.

If you are an experienced user, you can find tutorials that cover particular use cases, such as the Blue-Green Deployment tutorial on page 1984, or particular product features, such as how to use snapshots on page 1970.

Some tutorials provide a head-start by including resources that you can download, or that build on other tutorials. The JPetStore tutorial, for example, is used by several tutorials.

If you have topics that you would like covered by a tutorial, add a comment and describe your topic.

Basic tutorials

Tutorials for new users.

The tutorials in this section introduce the most-used product features. The HelloWorld tutorial on page 1915 is ideal for new users.

Create a simple helloWorld deployment

In this tutorial, you create a simple application with HCL DevOps Deploy (Deploy). You use standard plug-ins to create a component and define an application that deploys the component to an environment.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a component and add a user-defined property to it.
- Import artifacts to create a component version.
- Create a component process.
- Create an application and application process.
- Configure an environment with resources and apply a user-defined tag to it.
- Run the application process to deploy the component.

Time required:1 hour

Introduction

In this tutorial, you create a simple application with HCL DevOps Deploy (Deploy). You use standard plug-ins to create a component and define an application that deploys the component to an environment.

The deployment that you create in this tutorial, although simple, introduces you to the most-used product features. The `helloWorld` deployment that you create in the tutorial deploys a component by moving some files on the local file system to another location on the file system, presumably a location that is used by an application server.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a component and add a user-defined property to it.
- Import artifacts to create a component version.
- Create a component process.
- Create an application and application process.
- Configure an environment with resources and apply a user-defined tag to it.
- Run the application process to deploy the component.

Time required:1 hour

Time required

This tutorial takes approximately 1 hour to finish. If you explore other concepts that are related to this tutorial, it might take longer to complete.

System requirements

To run this tutorial, you must have an installation of Deploy.

Prerequisites

To complete this tutorial, you must have installed the Deploy server and at least one agent. Typically, the agent is installed on a computer or system other than the one where the server is located. However, for simplicity, the agent can be installed on the same system where the server is installed.

Configure the helloWorld tutorial system

To deploy the tutorial component, or any component, you must have an *agent*. An agent is a lightweight process that runs on a target computer and does the actual work of deploying components.

Before you begin

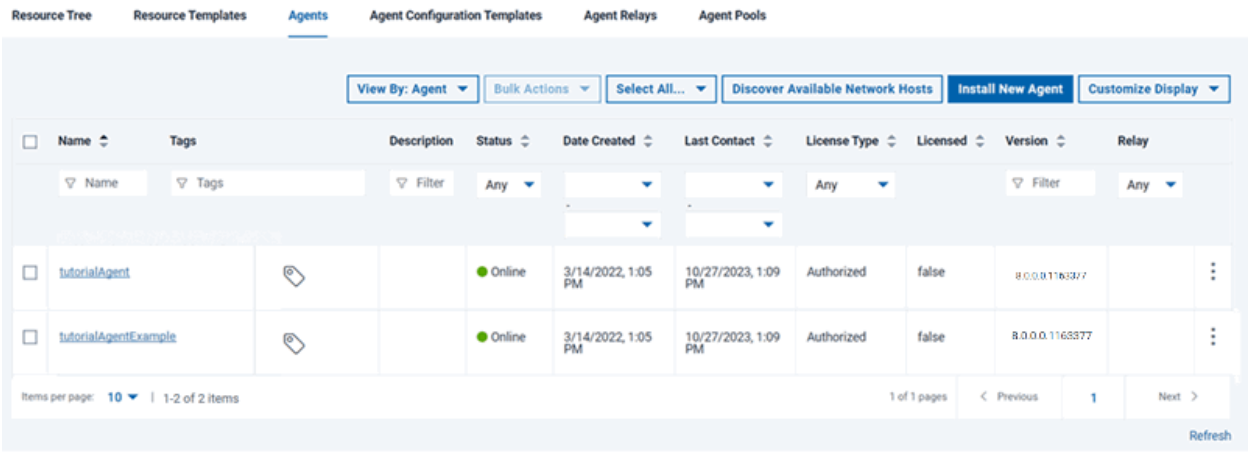
Before you begin the tutorial, make sure that you completed the Prerequisites on page 1916.

About this task

To complete the tutorial, you must have an agent that is running on a target computer system. This target computer can be a virtual image, or a physical computer, including the one where the server is installed. You must have access to this computer and be able to install the agent on it. In addition, you must create several directories on the file system where the server is located. These directories contain the artifacts that you deploy.

To prepare the tutorial system:

1. Provision a computer for use as the target computer.
2. Install an agent on the target computer.
   You can install the agent from the HCL DevOps Deploy (Deploy) web interface as described in Installing agents remotely. You can install the agent from the command line as described in Installing agents from the command line.
3. Designate the agent for importing versions.
   By default, one system-wide agent is designated to import component versions. If you skip this step or if another agent is already designated, you can still use the new agent when you import your component version. To designate the agent, see System settings on page 2086.
4. On the Deploy server, verify that the agent is installed by clicking **Resources > Agents**. Your agent must be included in the list of agents with a status of `Online`, as shown in the following figure:



5. Add the agent as a resource:

    a. To the left of the **Agents** tab, click the **Resource Tree** tab.

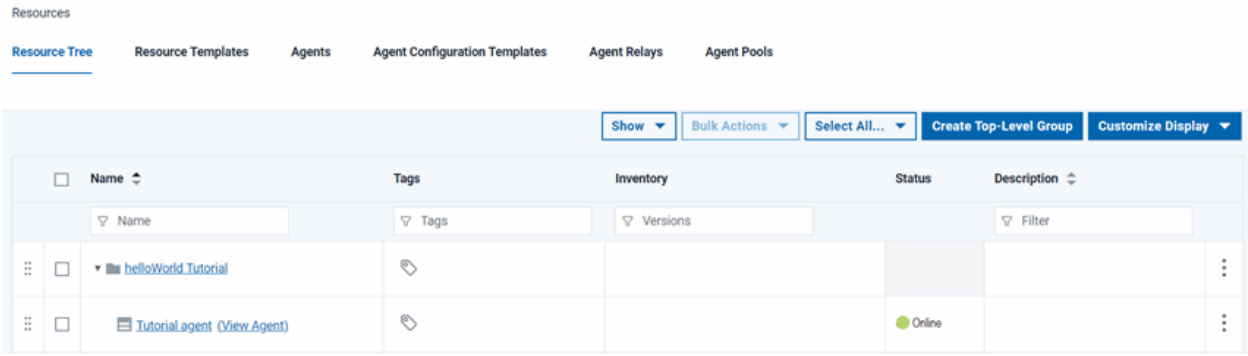
    b. Click **Create Top-Level Group**, give the new group a name such as `helloWorld Tutorial` and click **Save**.
    This group is a container for resources, such as agents and components.

    c. On the same line as the resource group, click ⋮ **Actions > Add Agent**, as shown in the following figure:

    d. In the **Create Resource** window, select the agent in the **Agent** list, and click **Save**.

Result

The agent is displayed as a child of the resource group.



6. On the computer where you installed the agent, add files that represent deployable artifacts to the file system.

    a. Create a directory and give it a name such as `helloWorld` on the file system where the agent is located.

    b. In the `helloWorld` directory, create a subdirectory that is named `1.0`.
    The source configuration the tutorial uses treats each subdirectory in the target directory as a different component version.

    c. In the `1.0` directory, place several files with any name and extension.
    The files represent the artifacts that the agent deploys by using the processes that you create. For the exercise, the number of files and their type is irrelevant, but, for best performance, limit their number and size.

    Note: If z/OS® is used on your target computer, convert text-type files that are used for artifacts to IBM®-1047 encoding.

Results

The server and agent are configured for the tutorial, and the files that represent artifacts are ready to use. If you installed the agent on a remote system, you must know the credentials and locations that you specified.

Lesson 1 Create the helloWorld component

Components contain *artifacts* and *processes*. Artifacts include runnable files, images, databases, configuration instructions, and anything else that is associated with a software project. Processes define the activities that components can perform.

Before you begin

Before you begin this lesson, make sure that you complete the steps that are described in Configure the helloWorld tutorial system on page 1916.

About this task

In many cases, artifacts are added to a component by connecting the HCL DevOps Deploy (Deploy) server to a computer system that hosts the artifacts. The server can import artifacts from build systems, source-code management systems, and file systems. Imported artifacts are stored in CodeStation, the artifact repository. The artifacts for the helloWorld component come from the file system where the Deploy agent is located. For more information about creating components, see Creating components on page 2248.

1. To create the helloWorld component:
    a. Click the **Components** tab and then click **Create Component**.
    In the window that opens, you define the component and specify the location of the artifacts for it.
    b. In the **Name** field, type `helloWorld`.
    c. In the **Source Configuration Type** list, select **File System (Versioned)**.
    This parameter defines the type of artifacts the component uses. All artifacts in a component share the source type. The File System (Versioned) type looks for the artifacts on the file system.

    Note: The None type can be used for configuration-type or operational-type deployments. For more information about deployment types, see Creating component processes on page 2425.

    d. In the **Base Path** field, specify the location of the `helloWorld` folder that you created earlier, such as `/home/user1/artifacts/shared/helloWorld`.
    The File System (Versioned) source type considers directories in the specified folder as representations of component versions. For this reason, make sure that you specify the `helloWorld` directory and not the `helloWorld/1.0` directory.
    e. Accept the default values for the other fields on the page.
    The **Preserve Execute Permissions** and **Import Versions Automatically** check boxes are cleared and the **Copy to CodeStation** check box is selected. The **Default Version Type** is set to **Full**.
    f. Click **Save**.
2. Create a component property.
    A property is user-defined parameter that can be set at design time or run time. Properties can be defined for components, processes, applications, environments, and other product elements. Here you set the property on the

component, but in situations where multiple components share the property, you might set the property value on the environment instead of setting it on every component.

a. On the "**Component: helloWorld**" page, click the **Configuration** tab and then click **Component Properties**.

b. Click **Add Property**.

c. In the **Edit Property** window, name the property something like `helloHome`.

d. In the **Value** field, type the name of the directory that you created to hold the component artifacts, for example `c:\helloWorld`.
The component process uses this property later to identify the target location for the component artifacts.

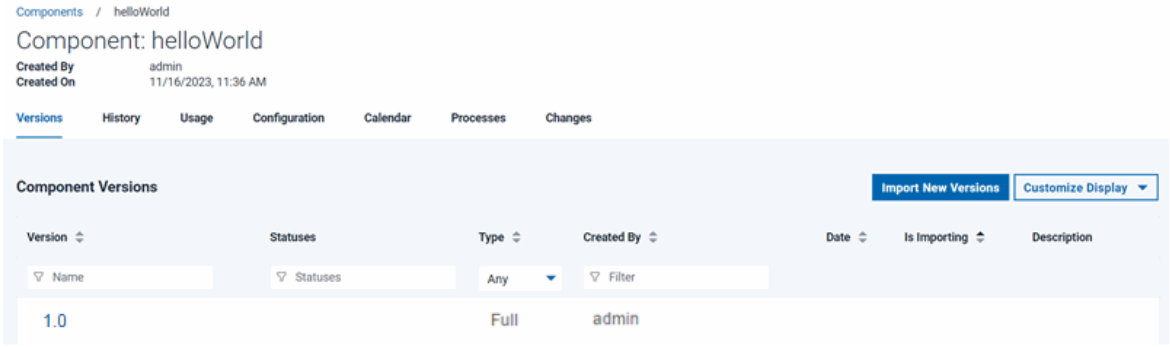
e. Click **Save**.

3. Import the component version:

a. Click the **Versions** tab.

b. Click **Import New Versions**.
Result
The server creates a version of the component that is based on the folder in the `helloWorld` folder, and imports the files in the `helloWorld/1.0` directory. In this case, the server creates a single version of the component that is based on the `helloWorld/1.0` folder. If other folders were in the `helloWorld` directory, they too would be imported and versions would be created from them.

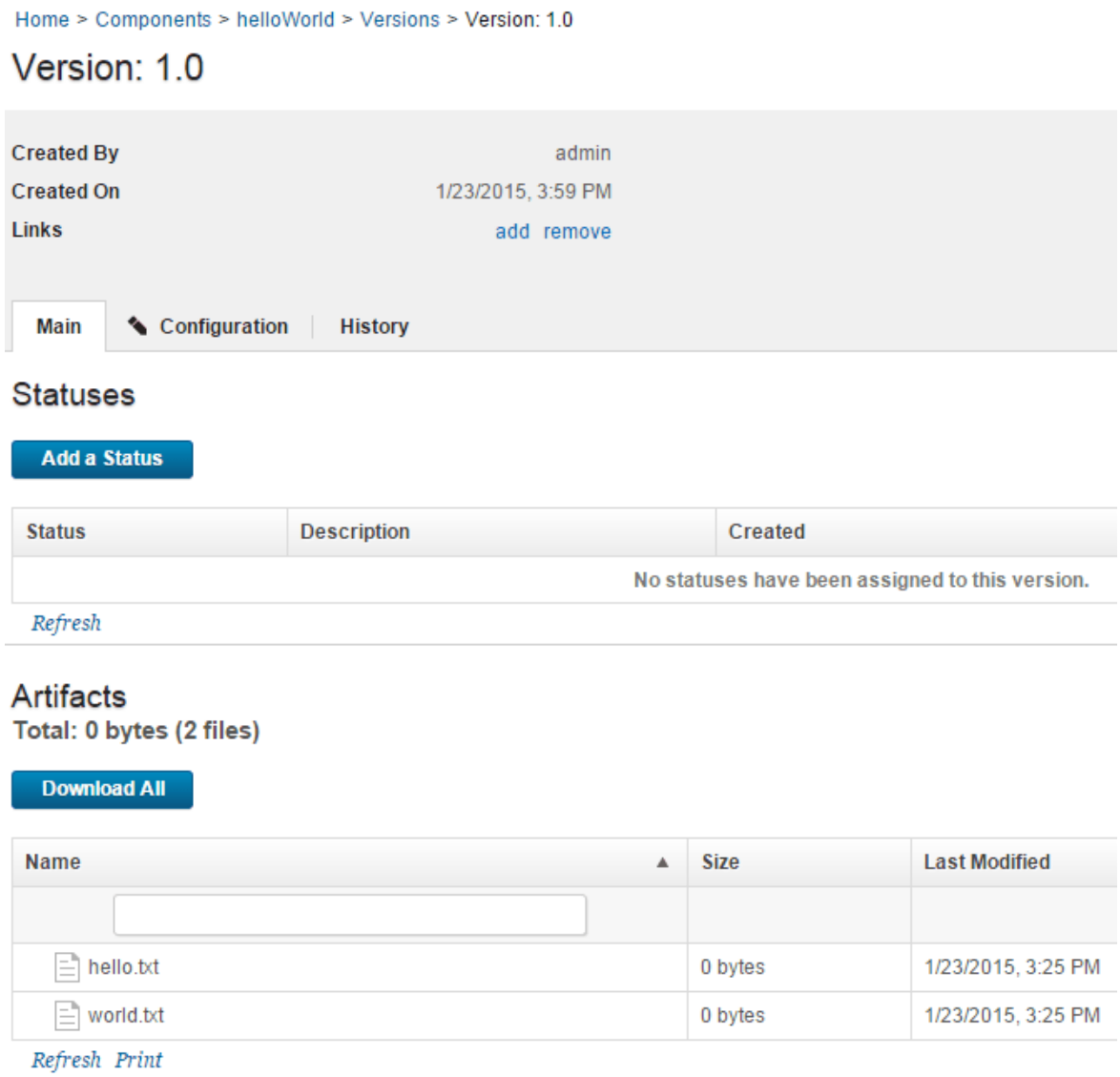
c. Verify that the list of versions includes version 1.0 of the component, as in the following figure:
Result



d. Click the version name, **1.0**.

e. Verify that the list of artifacts includes the files in the `helloWorld/1.0` directory, as in the following figure:

Result



Result

The helloWorld component is ready to use.

Results

In this lesson, you created a component that is based on files in the file system, and imported the initial version into CodeStation, the artifact repository. On the **helloWorld Component** page, you have the initial version, 1.0, listed.

What to do next

Component processes describe different ways to use a component, such as how to deploy it, install it, update it, or do other tasks with it. In the next lesson, you create a process that defines how to deploy the component. For more information about creating components, see Creating components on page 2248.

Lesson 2 Create a component process

Component processes are user-defined tasks that operate on components.

About this task

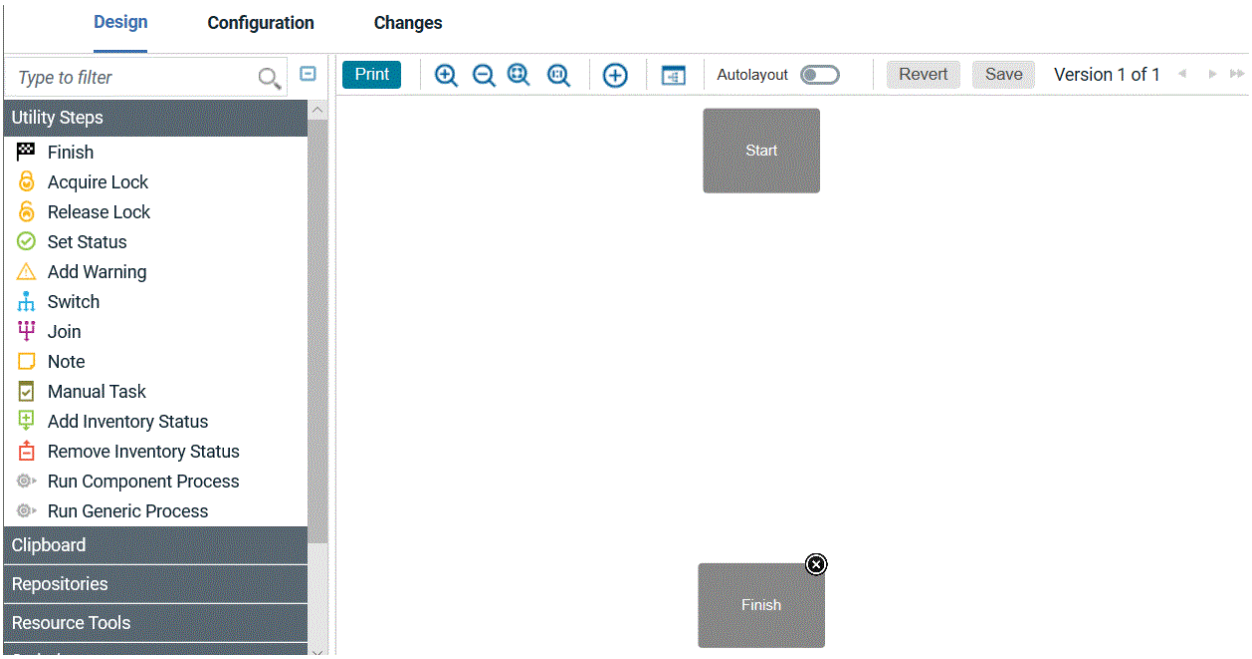
A component process is a succession of commands that are called *steps*. Steps can manipulate files, run system commands, set properties, pass information to other steps, and run programs. Steps are provided by *automation plug-ins*. Processes are designed with the drag-and-drop process editor where you drag plug-in steps onto the design editor and configure them as you go. Several plug-ins come with the product and others are available, which work with many different types of software. In this tutorial, you use two plug-ins, both of which come with the product: **FileUtils** and Deploy Versioned file storage. A component can have any number of processes defined for it, but a component must have at least one process.

In this lesson, you create a deployment process for the helloWorld component. Later, you create an application process that uses the component process to deploy the component.

To create the component process:

1. Define the component process:
    a. On the "**Component: helloWorld**" page, click the **Processes** tab, and then click **Create Process**.
    b. In the **Create Process** window, type `helloProcess` in the **Name** field.
    c. Accept the default values for the other fields, and click **Save**.
    The **Default Working Directory** field becomes the folder that the agent uses to do its work, such as downloading artifacts and creating temporary files. For the tutorial, the value becomes `\agent_install \var\work\helloWorld`.

    Result
    The process is listed on the **Processes** pane.
2. Open the **helloProcess** process editor and go to **Design**.
    Result
    The process editor lists plug-ins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and are automatically placed on the design area. You add steps to the process by dragging them onto the design area and arranging them between the **Start** and **Finish** steps.



3. Add a step that removes files from the working directory.
   To ensure that the agent works with the most recent files, remove files that remain from previous work:

   a. In the **Step Palette** pane, click **Utilities**.

   b. Click and drag the **Delete Files and Directories** step to the process editor.
      Result

      Click the edit properties icon , and the **Edit Properties** window opens, showing the properties for the step. Some of these properties are unique to the step, and others are the same for all steps.

   c. In the **Name** field, specify a name, for example, `Clean work space`.

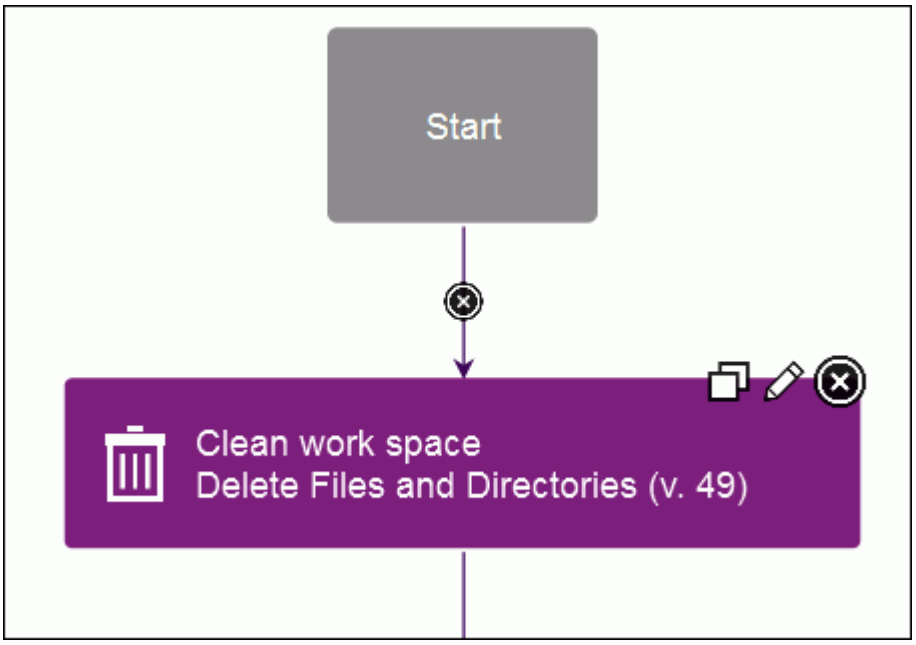
   d. In the **Base Directory** field, specify a single period (`.`).
      This value resolves to the folder that you specified as the agent working directory, such as *<DEPLOY_AGENT_HOME>*`/var/work`.

   e. Click the **Include** field, and in the **Edit Text** box, specify an asterisk (`*`); then, click **Save**.
      This parameter instructs the agent to remove all files in the working directory.

   f. Accept the default values for the other properties, and then click **OK**.

Result

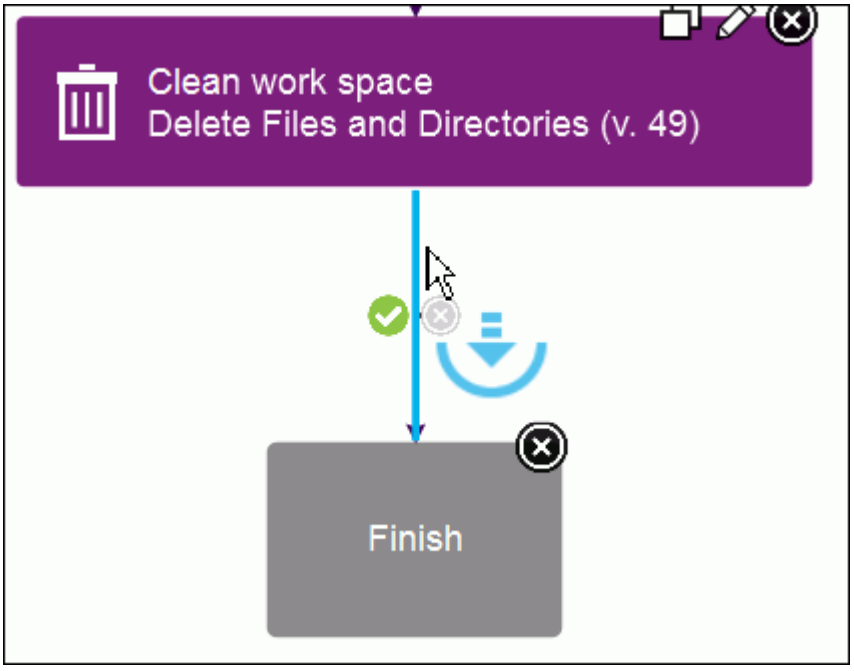
The new step is displayed in the design area. Each step has two icons at the upper-right corner of the step. Click the **Edit** icon to reopen the **Edit Properties** window where you can modify the step properties. Click the **X** icon to delete the step and connections to or from it.



The **Start** and **Finish** steps are automatically connected to the step.

4. **Optional:** If the process step became disconnected from the **Start** and **Finish** steps, connect the **Start** step to the **Clean work space** step. Beginning with **Start**, the connections between steps determine the process flow. You can define connections as you place steps or later.
5. Add a step to download the latest version of the component artifacts from the CodeStation repository.
   The **Download Artifacts** step is used in most deployment-type processes. This step downloads the specified version of the component artifacts to the agent's recently cleaned working directory. When you run the process, specify which version of the component artifacts to use.

a. Click **Repositories**, and under **Artifact/Deploy** drag the **Download Artifacts** step to the process editor. Be sure to highlight the connection arrow when you drag the step to the process editor so that connection arrows are automatically inserted from the previous step to the Finish step. See the highlighted connection arrow in this illustration:



b. Click the **Edit** icon , and in the **Edit Properties** window, accept the default values; then, click **OK**.

c. **Optional:** If you didn't highlight the connection arrow when you dragged the step to the editor, complete these steps:

   i. Delete the connection arrow between the **Clean work space** step and the **Finish** step.

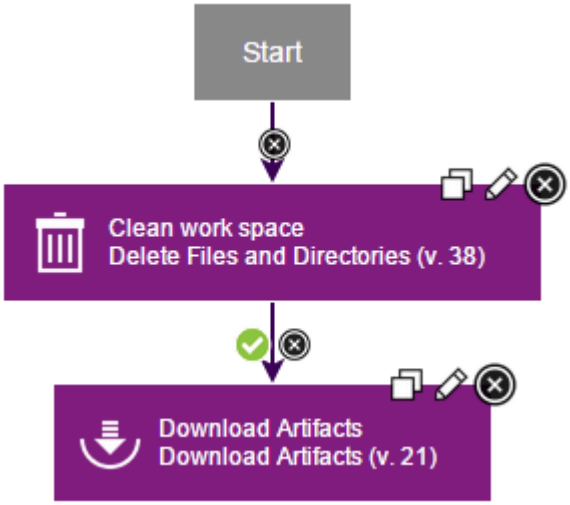
   ii. Draw a connection from the **Clean work space** step to the **Download Artifacts** step, and then connect the **Download Artifacts** step to the **Finish** step.

Tip: To manually connect steps, move the mouse over one step. The arrow icon is displayed, as the following figure shows. You use this icon to connect steps to one another.

. Click and drag the arrow icon over the subsequent step. The direction of the arrow defines the direction of the process flow.

Except for connections from **Start**, each connection has a *conditional flag*. The conditional flag represents the process status that is required before the connection can be followed. The connection is followed only if the originating step is complete, and the specified condition is met. The possible conditions are as follows: success, failure, or either. The default condition is success, which is represented by a green check mark, as

the following figure shows. In this example, the **Download Artifacts** step is started only if the **Clean work space** step finishes with a success condition. You can change the conditional flag by clicking the icon.



6. Add a step to create a compressed file.
   In this example, the compressed file contains the files that were downloaded in the previous step.

   a. Click **Utilities**, and under **FileUtils**, drag a **Create .zip File** step to the process editor.

   b. In the **Edit Properties** window, specify the name: `Compress artifacts`. (To open the **Edit Properties** window, the click the **Edit** icon .)

   c. In the **.zip File Name** field, type `helloWorld.zip`.

   d. In the **Base Directory** field, type a single period (`.`).
      The compressed file is created in the agent working directory.

      Note: If you specify the final destination directory instead of the agent working directory, the following step would be unnecessary. However, in addition to moving the file, the following step demonstrates how to use the property that you created earlier.

   e. In the text editor that opens for the **Include** field, type an asterisk (`*`); then, click **Save**.
      The compressed file contains all the files that were downloaded from the repository.

   f. Accept the default values in the other fields, and click **OK**.

   g. Verify that there is a connection from the **Download Artifacts** step to the **Compress artifacts** step. If the steps are not connected, manually connect them.

7. Add a step to move the artifacts.
   This step moves the compressed file to the target folder. For simplicity, the tutorial moves the file to a directory in the `\helloWorld` folder.

a. Expand **Utilities**, and under **FileUtils**, drag a **Move Directory** step to the process editor.

b. In the **Edit Properties** window, specify the name `Deploy artifacts`.

c. In the **Source Directory** field, type a single period (`.`).

d. In the **Destination Directory** field, type the following value: `${p:helloHome}/target`.
The `${p:helloHome}` value refers to the property that you created earlier. The `/target` directory is relative to the property value. If, for example, you defined the property value as `/home/user1/artifacts/shared/helloWorld`, the file is moved to `/home/user1/artifacts/shared/helloWorld/target`. For information about using properties, see Properties on page 2500.

e. In the text box that opens for the **Include Files** field, type: `*.zip`; then, click **Save**.

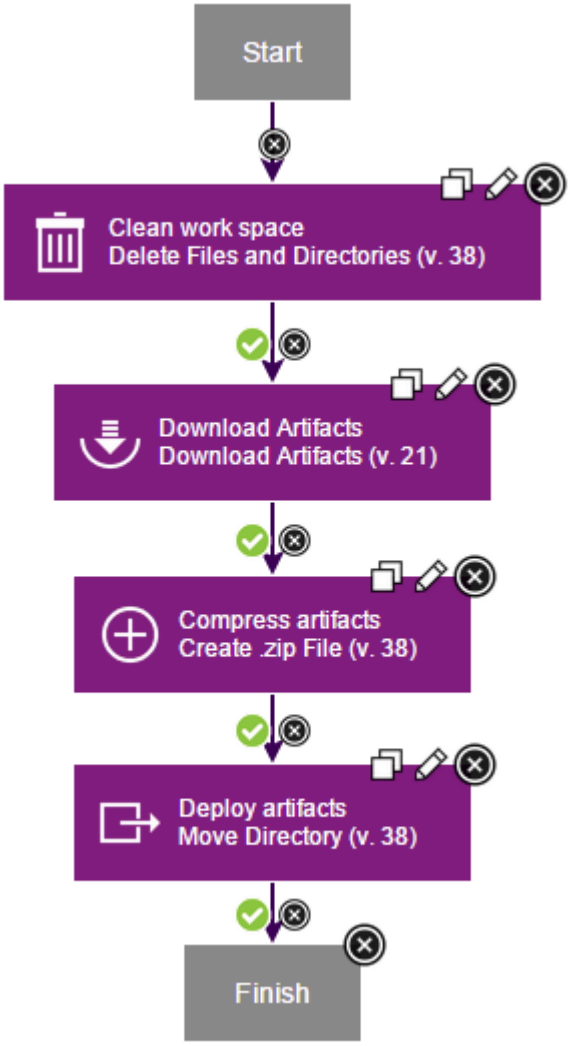
f. Accept the default values in the other fields, and click **OK**.

g. Verify that there is a connection from the **Compress artifacts** step to the **Deploy artifacts** step.

8. Verify that there is a connection from the **Deploy artifacts** step to the **Finish** step.
9. Save the process by clicking **Save** .

Results

The complete process for the helloWorld component looks like the following figure:



In this lesson, you created a component process to deploy the helloWorld component.

You can use component processes to automate many tasks on components. Later in this tutorial, you create an application process to deploy the component.

Lesson 3 Create an application

Applications manage components, typically by deploying them into environments.

About this task

To create an application, you identify the components that it manages; define at least one environment into which the components are deployed; and create a process to do the work. An environment maps components to agents and handles inventory, among other things. An application process is similar to but not identical to a component process. Application processes are primarily intended to direct underlying component processes and orchestrate multi-component deployments. In this lesson, you create an application and assign the helloWorld component to it.

1. Click the **Applications** tab, and then click **Create Application**.
2. Name the new application something like `hello Application`.
3. Accept the default values for the other fields, and click **Create**.
   For the tutorial, the default value for the **Notification Scheme** is fine. HCL DevOps Deploy (Deploy) integrates with LDAP and email servers, which enables the product to send event-based notifications. For example, the `default notification scheme` sends an email (if an email server is configured, see System settings on page 2086) when a deployment finishes. Notifications can also play a role in deployment approvals. See Managing Security on page 2149 for information about security roles.
4. Add the helloWorld component to the application:

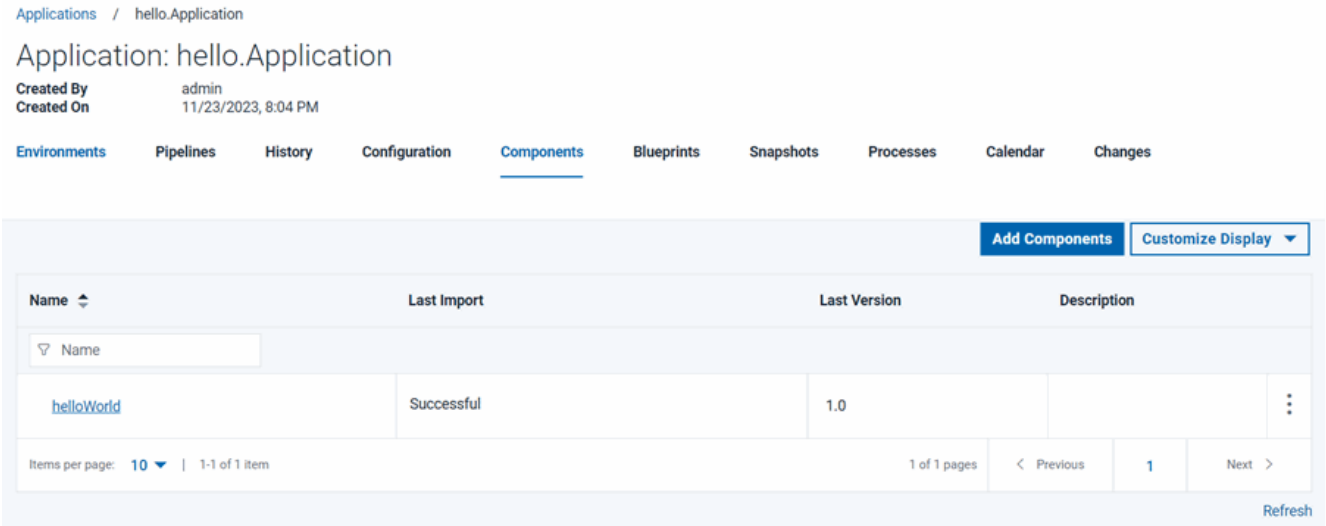
   a. From the **Application: hello Application** page, click the **Components** tab. This is the **Components** tab that is associated with the application, not the **Components** tab at the top of the page.

   b. Click **Add Component**.

   c. In the **Add a Component** window, select the **helloWorld** component, and then click **Save**.
   An application can deploy only components that are added to it.

Results

The application includes the helloWorld component, as shown in the following figure. Applications can have any number of components, and components can be included in any number of applications.



In this lesson, you created an application and assigned the helloWorld component to it. On the **Components** tab, you have the component that is managed by the application listed.

What to do next

For more information about creating applications, see Creating applications on page 2346.

Lesson 4 Configure an application environment

When you create an environment, you map resources to it that define where the parent application can run deployments.

About this task

An environment is a user-defined collection of resources that identify the components that can be deployed by the parent application, along with the agents that do the work.

1. Define the environment:
    a. On the **Application: hello Application** page, click **Environments**.
    b. Click **Create Environment**.
    c. Specify the name to be something like `helloDeploy`.
    d. Accept the default values in the other fields in this window, and click **Save**.
2. Add the resource group that contains the agent you created earlier to the environment.
   When you set up the agent, you added the agent as a resource to a resource group.

    a. Click the environment name.

    b. Using the **Resources** tab for the environment, click **Add Base Resources**.
    Result
    The **Add Resource to Environment** window shows all available resources.

    c. Select the check box next to resource group that you created earlier, `helloWorld Tutorial`, and then click **OK**.
    When you select the resource group, you automatically select the resources that are associated with it, such as the agent that you assigned to the group. Selecting an agent-type resource identifies the location, usually a computer, where deployments can occur.

3. Map the helloWorld component to this agent resource.

    a. As you hover the mouse over the row with the agent resource, click **Actions > Add Component**.

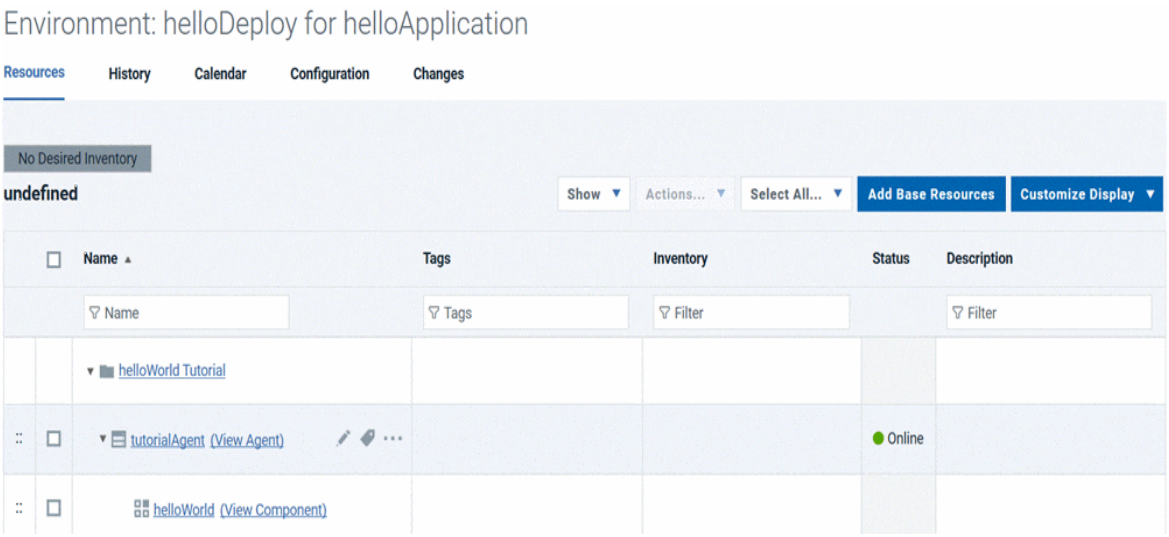
    b. Select the **helloWorld** component and then click **Save**.
    Only components added to the parent application are available to be mapped to the agent resource.

    Note: In order for a component to be deployed by an application, it must be added to the application and also mapped to an agent-type resource. A component that is added to an application but not mapped to an agent resource, cannot be deployed by that application. Similarly, a component that is mapped to an agent resource but not added to an application, cannot be deployed by that application.

4. Add a tag to the helloWorld component resource.
   User-defined tags can identify resources and be referenced in application processes.

    a. Click the check box for the **helloWorld** component.

    b. On the same row as the **Add Base Resources** push button, click **Actions > Add Tag**.
    The **Add Tag** selection is shown in the following figure:

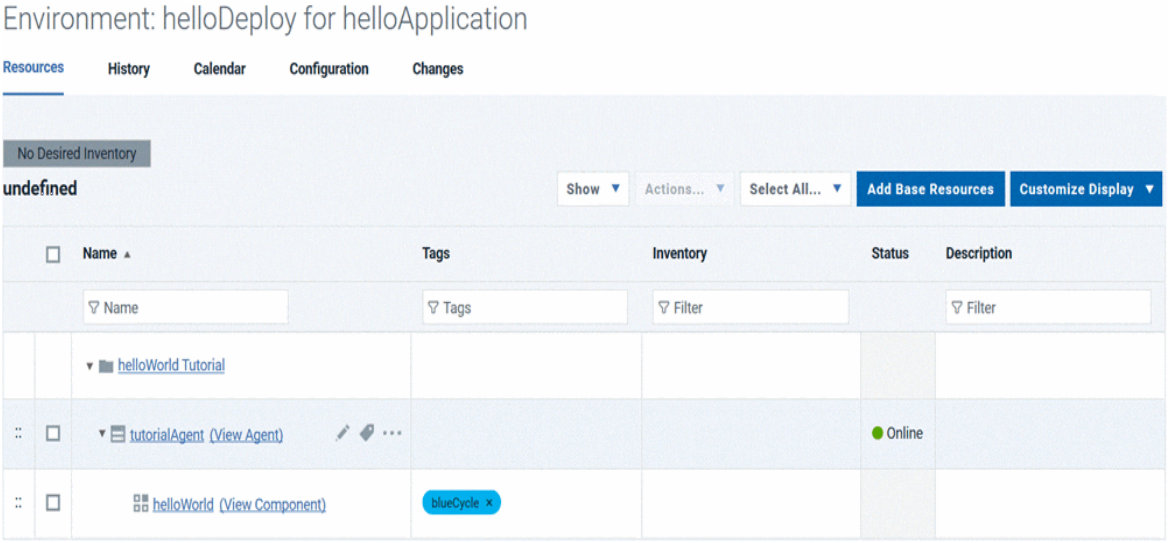


c. In the **Add New Tag** window, type something like `blueCycle` in the **Name** field, select a color for the tag, and then click **Save**.

While you can select any color, blue is a good choice. Applications can be configured to use resources with a specific tag. For example, a continually running environment might take one group of `blueCycle` tagged servers offline to update them, then put them back online and take another differently tagged group offline. You can also configure a process to deploy all components with a particular tag.

Result

The finished resource with the `blueCycle` tag is shown in the following figure:



Results

The new environment is displayed on the **Environments** tab. After the environment is prepared, you can create the application process.

In this lesson, you created an environment and added a resource group to it, then added a tag to the resource.

On the **Resources** tab, you have the resource group with your agent and the `helloWorld` component.

For more information about creating environments, see Application environments on page 2354.

Lesson 5 Create an application process

Application processes direct underlying component processes and orchestrate multi-component deployments.

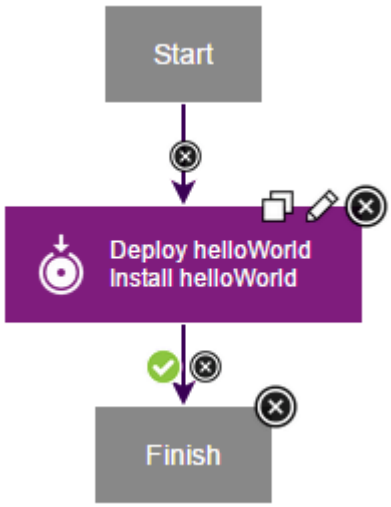
About this task

An application process, like a component process, consists of steps that are configured with the process editor. In this lesson, you create an application process that installs the helloWorld component by calling the component process that you created earlier. You must use application processes to deploy components.

1. Click the **Applications** tab, and then click **hello Application**.
2. Click **Processes**, and then click **Create Process**.
3. In the **Create an Application Process** window, name the new application process something like `hello App Process`.
4. Accept the default values for the other fields, and click **Save**.
5. To open the new process in the process editor, click the new process.
6. Add a step that deploys the helloWorld component:
    a. From the list of steps, add an **Install Component** step to the design area, and then on the process step click the **Edit** icon .
    b. In the **Edit Properties** window, name the step something like `Deploy helloWorld`.
    c. In the **Component** list, select the **helloWorld** component.
    All components that are associated with an application are available.
    d. In the **Component Process** list, select the **helloProcess** component process.
    If multiple processes were defined for the helloWorld component, they are listed.
    e. In the **Limit to Tag** list, select the **blueCycle** tag that you defined earlier.
    After you select **blueCycle**, the application deploys only components with this tag.
    f. Accept the default values for the other fields, and click **OK**.
7. Verify that the **Start** step is connected to the **Deploy helloWorld** step.
8. Verify that the **Deploy helloWorld** step is connected to the **Finish** step.
9. Save the process by clicking **Save**.

Results

The complete application process looks like the following figure:



Example

In this lesson, you created an application process that deploys the helloWorld component.

The **Processes** tab lists the application process.

What to do next

For more information about creating application processes, see Application processes on page 2428.

Lesson 6 Deploy the component

To deploy the component, run the application process on an environment.

About this task

Using the application environment and process that you created in previous steps, you can deploy the component.

1. Open the application page by clicking **Applications** and then clicking the application name.
2. In the same row as your environment, click the **Request Process** icon .
   The **Run Application Process** window opens.
3. In the **Process** list, select the **hello App Process** process.
4. Accept the default value for the **Only Changed Versions** parameter.
   If this check box is selected, no previously deployed component versions are deployed.
5. Specify component version by clicking **Select For All**, and then select **Latest Available** from the **Choose Component Versions** tab or specify a snapshot from the **Select a Snapshot** tab.
6. Set the date, time, and pattern of deployment. The **Run Now** option is enabled by default for immediate deployment. You can schedule deployments by disabling this option.
7. Review your deployment configuration and make changes, if necessary.
8. Click **Submit Deployment** when ready.
   Result

The **Application Process Request** page shows the progress of the request. From here, you can watch as the process runs.

If the application process execution is not successful, expand a process step and click **View Child Execution** to open the Deployment of Component pane, which can be used to review the deployment details. The **Deployment of Component** page shows the log for each step in the component process. In the Deployment of Component pane, you can review the deployment output log, error log, and input and output parameters.

Results

The application process installed the helloWorld component into the target environment.

Application processes can also uninstall or update components or run other configuration.

What to do next

For more information about running deployments, see Deploying applications on page 2526.

Summary

In this tutorial, you learned how to deploy a component with HCL DevOps Deploy (Deploy).

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How components, applications, and environments work together.
- How to create component versions.
- How to use plug-ins to create processes.
- How to configure environments for deployments.
- How to create and use properties.
- How to manage resources and agents.
- How to create tags and use them in applications.
- How to run deployments.

Additional resources

For more information about Deploy, see the help system and related resources on Deploy websites.

Intermediate tutorials

Tutorials for experienced users.

The tutorials in this section are for users who want to explore product features in more depth. The JPetStore on page 1934 tutorial shows you how to deploy a data-driven web application. In addition to providing more in-depth topics, the JPetStore tutorial is referenced by some of the advanced tutorials on page 1984.

Deploying a simple web application

In this tutorial, you deploy a simple web application with HCL DevOps Deploy (Deploy). You create components, create an application that contains those components, and then deploy the components to an environment.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Import artifacts from a build to create a component in Deploy.
- Create component processes and application processes to automate deployment tasks.
- Configure environments on which to deploy applications.
- Run application processes to deploy the application components.
- Update an application with new artifacts.

Time required

2-3 hours

Introduction

In this tutorial, you deploy a simple web application with HCL DevOps Deploy (Deploy). You create components, create an application that contains those components, and then deploy the components to an environment.

Deploy helps you to plan and automate the deployment of complex applications to development, test, and production environments. It can deploy applications as often as needed, helping you to find problems early in the release cycle and provide predictability late in the release cycle.

The web application that you deploy in this tutorial is a simple three-tiered web application with a database, web interface, and application logic. You import these three components to the Deploy server and create a single application that deploys these three parts. Then, you create automation instructions that are called *processes* that describe how to deploy each component. Finally, you use those processes to deploy the components automatically to a target system.

In later lessons, you update the components to a new version. Working with the sample application in this way demonstrates how you can manage a multipart application with Deploy.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Import artifacts from a build to create a component in Deploy.
- Create component processes and application processes to automate deployment tasks.
- Configure environments on which to deploy applications.
- Run application processes to deploy the application components.
- Update an application with new artifacts.

Time required

This tutorial takes 2 - 3 hours to finish. If you explore other concepts that are related to this tutorial, it could take longer to complete.

System requirements

To run this tutorial, you need the following resources:

- An installation of Deploy, version 8.2 or later
- Two computers, virtual images, or cloud instances to use as agents for the Deploy server

Prerequisites

- Make sure that you have access to the Deploy server. For information about installing a server, see Installing the DevOps Deploy server.
- The server must have the Apache Tomcat, DBUpgrader, and FileUtils plug-ins installed. If you do not have access permissions to install plug-ins, ask the system administrator to install these plug-ins. For instructions, see Installation of plug-ins on page 2558. You can verify that the plug-ins are installed by clicking **Settings > Automation > Installed Plugins** and finding Tomcat, DBUpgrader, and FileUtils in the list of installed plug-ins. The FileUtils plug-in is installed by default.
- You must have an account on the server with the permissions to create and edit applications, environments, and components, and to create and run processes.

Configuring the agent and target system

To deploy applications with HCL DevOps Deploy (Deploy), you need one or more *agents*. The server uses the agents to deploy the application components on target systems.

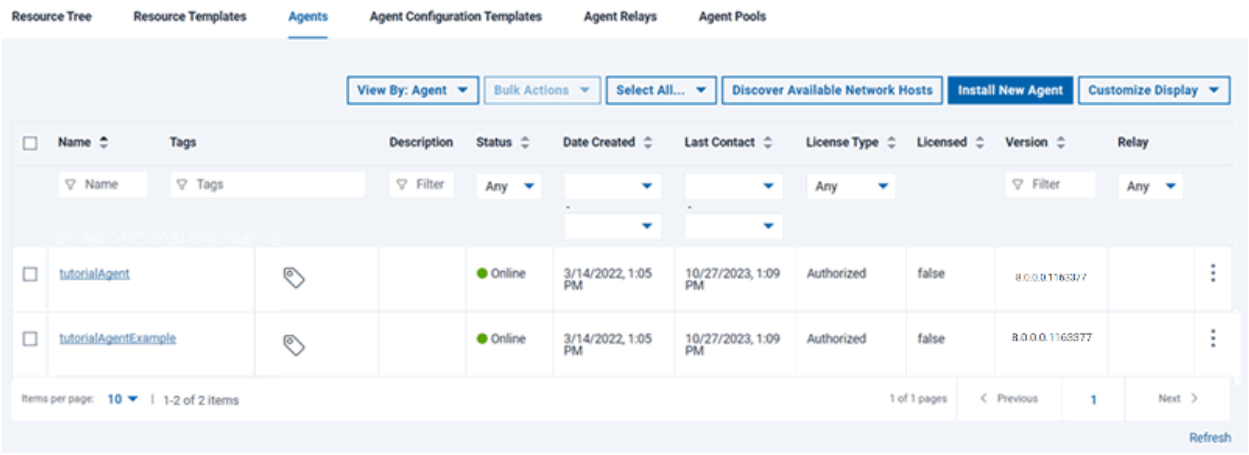
Before you begin

Before you begin the tutorial, make sure that you completed the Prerequisites on page 1935.

About this task

To work through this tutorial, you need an agent that is running on a target computer system. This target computer can be a cloud resource, a virtual image, or a physical computer. You must have access to this computer and be able to install MySQL/MariaDB, Apache Tomcat, and the Deploy agent on it. Follow these steps to set up an agent and a target system for your work with the tutorial:

1. Prepare a computer for use as the target system.
2. Install an agent on the target system.
   See Installing agents.
3. On the Deploy server, verify that the agent is installed by clicking **Resources > Agents**. Your agent must appear in the list of agents with a status of **Online**, as shown in the following figure:

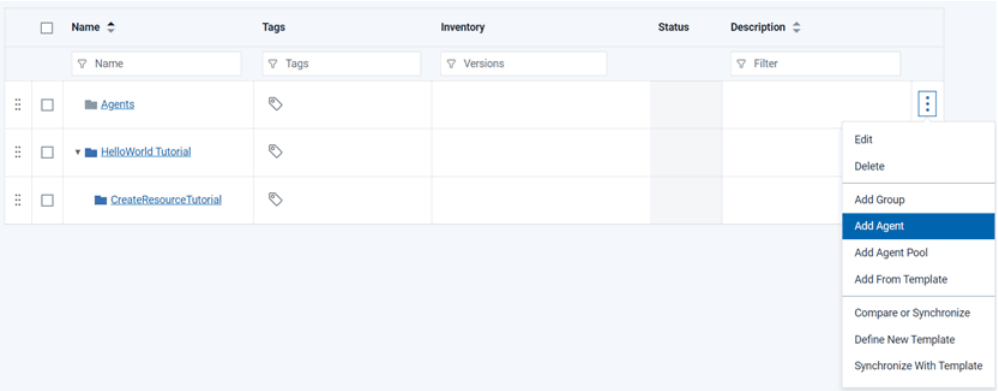


4. Add the agent as a resource:

    a. Go to the **Resource Tree** tab.

    b. Click **Create Top-Level Group**, give the new group a name such as `JPetStore agents` and click **Save**.
    This group is a container for resources, such as agents.

    c. On the same row as the new resource group, click **Actions > Add Agent**, as shown in the following figure:



    d. In the **Create Resource** window, select the agent in the **Agent** field, give a name to the agent, and click **Save**.

Result

The agent is now listed as a child of the resource group. You must add agents to resource groups in this way so that applications can access the agents through the resource group.

5. Install and start the MySQL/MariaDB server on the target system.
The application uses a MySQL/MariaDB database.

Example

If you are using a Linux™ operating system, you can use the package manager to install and start MySQL/MariaDB. For example, on Red Hat Enterprise Linux™ version 7, run the following commands:

```
yum install mariadb mariadb-server
```

```
systemctl enable mariadb
```

```
systemctl start mariadb
```

Example

On Windows™, you can use a graphical installer or download a binary package for the database server. Then, start the server by going to the `bin` folder and running the mysqld command.

6. Create an empty database for the application to use:

    a. Open a command-line window and log in to the database as the root user.

    Example

    For example, run this command from the command line:

    ```
    mysql -u root
    ```

    Result

    The command line now shows a prompt that looks like this prompt:

    ```
    MariaDB>
    ```

    b. Create the database with the following command:

    ```
    create database jpetstore;
    ```

    Result

    If the command runs properly, you see the message `Query OK, 1 row affected`.

    c. Create the database user with the following command:

    ```
    create user 'jpetstore'@'localhost' identified by 'jppwd';
    ```

    Result

    If the command runs properly, you see the message `Query OK, 0 rows affected`.

    d. Give the user access rights to the database with the following command:

    ```
    grant all privileges on jpetstore.* to 'jpetstore'@'localhost';
    ```

    Result

    If the command runs properly, you see the message `Query OK, 0 rows affected`.

    e. Type `quit` to exit the MySQL command line.

7. Verify that the user can access the database:

    a. From the command-line window, log in to the database as the new user:

    ```
    mysql -u jpetstore -jppwd
    ```

    b. Run the following command to list the databases on the system:

    ```
    show databases;
    ```

    c. Verify that the `jpetstore` database appears in the list, as shown in the following figure:

    Result

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jpetstore          |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

d. Type `quit` to exit the database command line.

8. Install Apache Tomcat on the target system.
   You can use Tomcat version version 9.
   Example
   A simple way to install Tomcat on a Linux™ operating system is to download and install the binary distribution of Tomcat, as in the following steps:

   a. Download the binary distribution of Tomcat, such as `apache-tomcat-9.0.100.tar.gz` .

   b. Extract this file to a folder on the target computer, such as `/opt/apache-tomcat-9.0.100` .
      Example
      For example, on a Linux™ operating system you might run the following command:

      ```
      tar -xf apache-tomcat-9.0.100.tar.gz ; mv apache-tomcat-9.0.100/ /opt
      ```

9. Add an administrative user to the Tomcat server in the `conf/tomcat-users.xml` file.
   This user account is necessary to deploy applications to Tomcat.
   Example
   For Tomcat version 9, open the file `conf/tomcat-users.xml` and uncomment the `role` and `user` tags at the bottom of the file. Remove the text `<must-be-changed>` and replace it with passwords for the default user accounts. If you do not remove this text, the configuration file is unreadable and your sample application will not be able to access Tomcat. Then, add the following lines of code to those tags:

   ```
   <role rolename="manager-script"/>
   <role rolename="manager-gui"/>
   <user username="tomcatmanager" password="tomcatmanager" roles="manager-script,manager-gui"/>
   ```

   The end of the file might look like the following example:

   ```
   <role rolename="tomcat"/>
     <role rolename="role1"/>
     <user username="tomcat" password="tomcat" roles="tomcat"/>
     <user username="both" password="tomcat" roles="tomcat,role1"/>
     <user username="role1" password="tomcat" roles="role1"/>

     <role rolename="manager-script"/>
     <role rolename="manager-gui"/>
     <user username="tomcatmanager" password="tomcatmanager" roles="manager-script,manager-gui"/>
   ```

```
</tomcat-users>
```

10. In the `conf/server.xml` file, modify the default HTTP ports to avoid conflicts.

    a. Replace all instances of port number `8080` with port number `8081`.

    Example

    To update the port numbers from the command line, run this command:

    ```
    sed -i 's/8080/8081/g' /opt/tomcat_directory/conf/server.xml
    ```

    b. Replace all instances of port number `8443` with port number `8445`.

    Example

    To update the port numbers from the command line, run this command:

    ```
    sed -i 's/8443/8445/g' /opt/tomcat_directory/conf/server.xml
    ```

Results

Now the agent and the target system are configured for your work with the sample application. You need the passwords and locations that you specified later.

Lesson 1: Adding components

Components are groups of deployable artifacts that make up an application. Components include runnable files, images, databases, and configuration instructions.

Before you begin

- Make sure that your HCL DevOps Deploy (Deploy) server has the prerequisites that are listed in Prerequisites on page 1935.
- Make sure that you have set up the agent for the sample application as described in Configuring the agent and target system on page 1935.

About this task

These steps describe how to create the components of the sample application.

The JPetStore sample application includes three components:

- The application component includes the logic of the application. This component consists of a single web archive file that is named `JPetStore.war`.
- The web component includes the static web content for the application, including images for the web interface.
- The database component includes the database for the application and scripts that upgrade the database schema to new versions.

In most cases, you add components by connecting the Deploy server to the system that hosts the artifacts. The server can import artifacts from many build systems and source-code management systems. The server can create new versions of the components automatically as the artifacts change or as new builds run. However, for simplicity, in this lesson you copy the

files for the components to the server and access them from there. For more information about creating components from artifacts on build systems and source-code management systems, see Creating components on page 2248.

1. Download the `artifacts.zip` file by clicking the following link:
   artifacts.zip.
2. Copy the `artifacts.zip` file to the Deploy agent computer and extract it.

   This file contains a folder named `shared` which contain separate folders for each of the components; these folders are named `app`, `web`, and `db`. Each of these folders contains one or more folders with names such as `1.0` or `1.1`; these folders represent different versions of the artifacts.

   Notes:
   - For the tutorial, these folders must be on the same system that hosts the Deploy agent.
   - Note the locations of these folders as you will need the locations later.
   - Ensure that the JARs in the `db` folder are compatible with the version of MySQL/MariaDB database installed on the agent computer.

3. Log in to the Deploy server.
4. Create the application component:

   a. Click the **Components** tab, and then click **Create Component**.
   In this window, you specify where the artifacts for the component are.

   b. In the **Name** field, type `JPetStore-APP`.

   c. In the **Source Configuration Type** list, select **File System (Versioned)**.

   d. In the **Base Path** field, specify the location of the `app` folder on the server, such as `/home/user1/artifacts/shared/app`.

   e. Make sure that the **Copy to CodeStation** option is selected to copy the artifacts from the given source into the Deploy server's CodeStation repository.

   f. Under **Default Version Type**, click **Import new component versions using a single agent**, and then select your agent in the **Agent for Version Imports** list

   g. Accept the default values for the other fields on the page.

   h. Click **Save**.

5. Import the versions of the component:

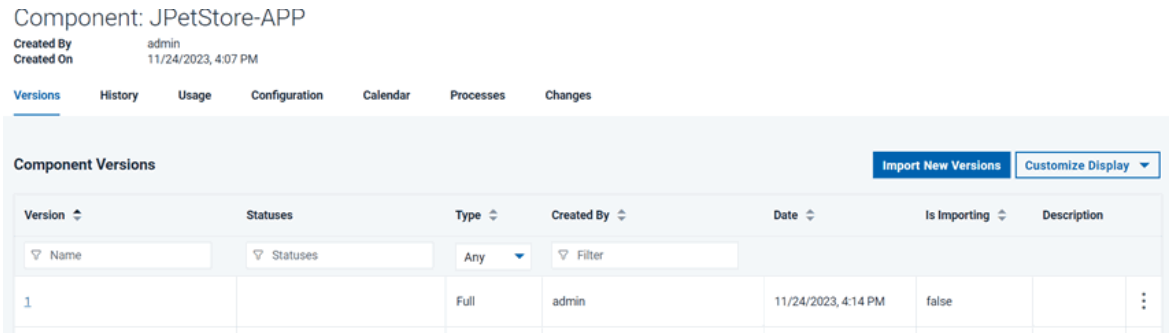
   a. Click the **Versions** tab.

   b. Click **Import New Versions**.
   Result

The import process is listed in the **Currently Running Version Imports** section. Then the process is finished, the server creates versions of the component based on the folders in the app folder. In this case, the server creates a single version of the component that is based on the `app/1.0` folder. Importing versions of small components like this one typically takes a few seconds. You might need to refresh the page or click the **Refresh** link at the bottom of the table to see the new version.

c. Verify that version 1.0 of the component is included in the list of versions, as in the following figure:

Result



If the version import failed, make sure that the files are available on the agent system and that the location of these files is correct on the component **Configuration** tab.

Result

The component is available to be used in one or more applications.

6. Create the database component:
   a. Click the **Components** tab, and then click **Create Component** again.
   b. Specify the name to be `JPetStore-DB`.
   c. In the **Source Configuration Type** list, select **File System (Versioned)**.
   d. In the **Base Path** field, specify the location of the db folder on the server, such as `/home/user1/artifacts/shared/db`.
   e. Under **Default Version Type**, click **Import new component versions using a single agent** and then select your agent in the **Agent for Version Imports** list
   f. Accept the default values for the other fields on the page.
   g. Click **Save**.
   h. Click **Versions** and then click **Import New Versions**.

Result

Now the server shows two versions of the database component.

7. Create the web component:
   a. Click the **Components** tab and then click **Create Component** again.
   b. Specify the name to be `JPetStore-WEB`.
   c. In the **Source Configuration Type** list, select **File System (Versioned)**.
   d. In the **Base Path** field, specify the location of the web folder on the server, such as `/home/user1/artifacts/shared/web`.

e. Under **Default Version Type**, click **Import new component versions using a single agent** and then select your agent in the **Agent for Version Imports** list
f. Accept the default values for the other fields on the page.
g. Click **Save**.
h. Click **Versions** and then click **Import New Versions**.

Result

Now the server shows two versions of the web component.

8. So that you can try updating the components later, delete the newest versions of the database and web components:

a. Click **Components** and then click the `JPetStore-DB` component.

b. Click **Versions**.

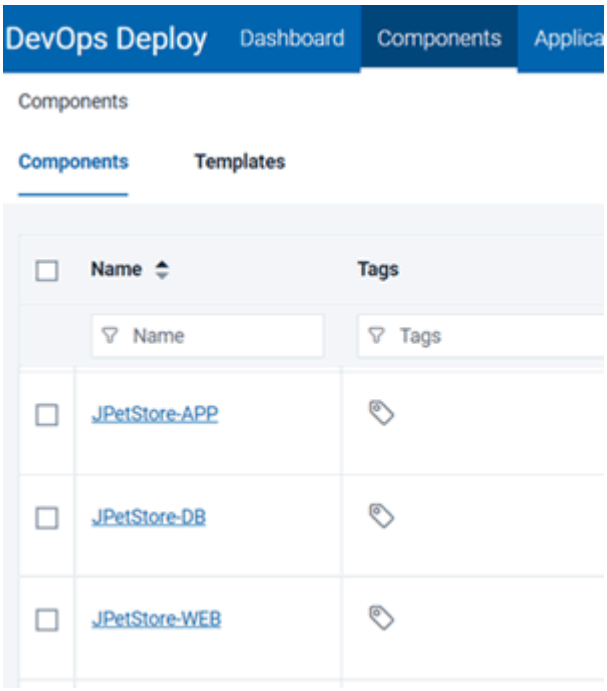
c. In the list of versions, in the same row as the version that is named `1.1`, click **Delete** and then click **OK**.
You will import this version again later.

d. Similarly, delete version `1.1` of the `JPetStore-Web` component.

Results

In this lesson, you added components that are based on files in the file system. In a production scenario, you would connect the component to a source-code management system such as Subversion or a build system such as Jenkins.

On the **Components** page, you now have three components, each with one version, as shown in the following figure:



In a later lesson, you add these components to an application.

What to do next

In addition to files and other artifacts, components also contain *component processes*, which describe how to deploy, install, uninstall, update, or run other tasks on a component. In the next lesson, you create processes that describe how to install the components.

Lesson 2: Creating component processes

Component processes describe automated tasks that run on components. Component processes can deploy, install, uninstall, update, or run other tasks on a component.

About this task

There are three types of processes:

- Generic processes run outside the context of components or applications. Generic processes are shown on the top-level **Processes** tab. You can use generic processes in these instances, for example:
  - In conjunction with a number of different component processes that do not fit well into a component template
  - When you want to run processes that directly affect HCL DevOps Deploy (Deploy) agents that are unrelated to component, such as changing the agent configuration
- Application processes run within the context of applications. In many cases, application processes call component processes. For example, an application process can call the component processes that deploy their associated components.
- Component processes run tasks on a single component, such as deploying it, uninstalling it, or running configuration tasks on it.

Processes are lists of steps and connections between those steps. Each step is an individual command that runs on a target computer. Steps can manipulate files, run system commands, download files, and run programs. Plug-ins contribute most of these steps; plug-ins are available that work with many different types of software.

Each component must have at least one component process to deploy or install the component. In this lesson, you create a deployment process for each of the three components. Later, you create an application process that calls these component processes to deploy each component.

Creating a component process for the web component

The component process for the web component downloads the most recent version of the component artifacts and puts those artifacts in the correct folder.

About this task

Follow these steps to create the component process that deploys the web component:

1. On the **Components** tab, click the **JPetStore-WEB** component.
2. Click **Processes**, and then click **Create Process**.
3. In the **Create Process** window, specify the name to be `Deploy component`.
4. In the **Process Type** list, select **Deployment**.
   This list has other options for processes that uninstall or configure components.

5. Accept the default values for the other properties, and click **Save**.
   Result
   The process opens in the process editor. This editor shows the steps in the process in a graphical form, as a flowchart. The **Start** and **Finish** boxes represent the beginning and the end of the process. From here, you add steps to the process and link them between the **Start** and **Finish** steps to show the order of the steps.
6. Add a step to clean any files out of the working directory.
   The process runs within a working directory. To ensure that you are using the most recent version of the files in the web component, run a command to clean the working directory:

   a. At the left of the process editor, in the step palette, expand **Utilities**.
      The step palette menu shows the available steps. You can explore the palette categories to find steps, or you can type in the search box at the top of the window.

   b. Under **FileUtils**, click and drag the **Delete Files and Directories** step to the process editor.
      Result
      The step is inserted between the **Start** and **Finish** steps. The inserted step is automatically connected to these steps **Edit Properties** window opens, showing the properties for the step. Some of these properties are unique to the step and others are the same for all steps.

   c. Click the **Edit** icon , and the **Edit Properties** window opens. In the **Name** field, specify the name to be `Clean working directory`.

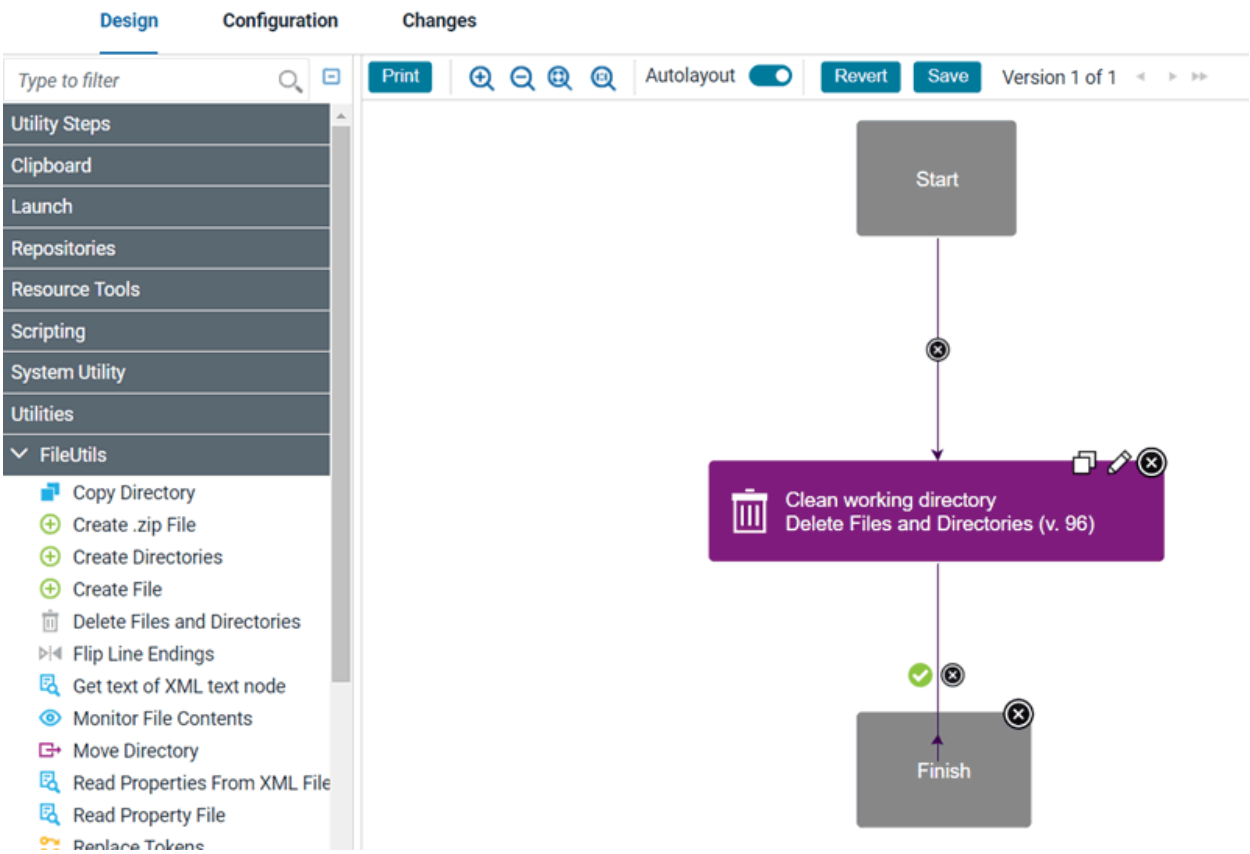
   d. In the **Base Directory** field, specify a single period (`.`).

   e. In the **Include** field, specify the following code:

      ```
      **/*
      ```

   f. Accept the defaults for the other properties and then click **OK**.

   Result
   The new step is displayed as a box in the process editor. The process editor looks like the following figure:



7. Add a step to download the latest version of the component artifacts.
   The **Download Artifacts** step is used in most deployment processes. This step downloads the specified version of the component artifacts to the target computer. When you run the process, you will specify whether to use the most recent version of the component artifacts or a specific version.

   a. **Tip:** When you drag the first step into a process editor, that step is automatically connected to the **Start** and **Finish** steps. When you drag subsequent steps into the process editor, place the cursor in the target position so that the connection arrow is highlighted, and then place the step. The new step is automatically connected. You can always change step locations and manually connect steps later.

   In the step palette, expand **Repositories**, and under Artifact/Deploy, drag the **Download Artifacts** step to the process editor.

   b. Click the **Edit** icon, and in the **Edit Properties** window, accept the default values; then, click **OK**.

8. Add a step to remove the old web content from the server:

   a. Expand **Utilities**, and under FileUtils, drag another **Delete Files and Directories** step to the process editor.

   b. In the **Edit Properties** window (click the **Edit** icon), and specify the name as `Remove old content`.

   c. In the **Base Directory** field, specify the following directory:

```
webapps/${p:environment/tomcat.contextroot}
```

The variable represents the location where Tomcat stores your deployed files. You specify a value for this variable later.

d. In the **Include** field, specify the following directory:

```
images
```

e. In the **Working Directory** field, specify the following variable:

```
${p:environment/tomcat.home}
```

This variable represents the main folder of the Tomcat web server. You will specify a value for this variable later.

f. Accept the default values in the other fields, and click **OK**.

9. Add a step to deploy the new content to the server.
This step copies the component artifacts to the application server.

a. Expand **Utilities** and under **FileUtils**, drag a **Move Directory** step to the process editor.

b. In the **Edit Properties** window, specify the name to be `Deploy new content`.

c. Click the **Edit** icon, and in the **Source Directory** field, specify a single period (`.`).

d. In the **Destination Directory** field, specify the following directory:

```
${p:environment/tomcat.home}/webapps/${p:environment/tomcat.contextroot}/
```

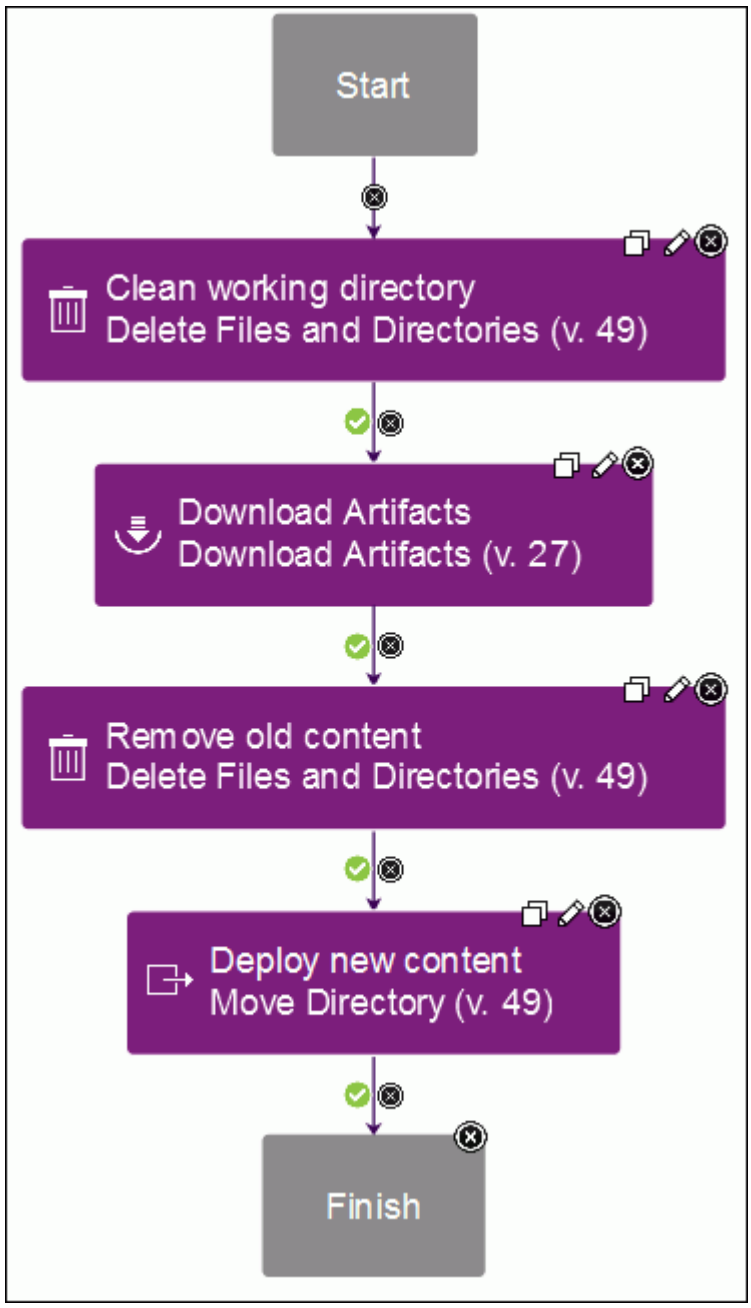
e. In the **Include Files** field, specify the following code:

```
images/*
```

f. Accept the default values in the other fields and click **OK**.

Result

Now the process has four steps, as in the following figure:



Each step has three icons at the upper right corner of the step. The **Edit** icon opens the **Edit Properties** window. The X icon deletes the step. The Copy icon copies the step to the **Clipboard** folder so you can duplicate it.

Also, if you move the mouse over a step, an arrow icon is displayed over the center of the step. This icon is used to manually create links between steps.

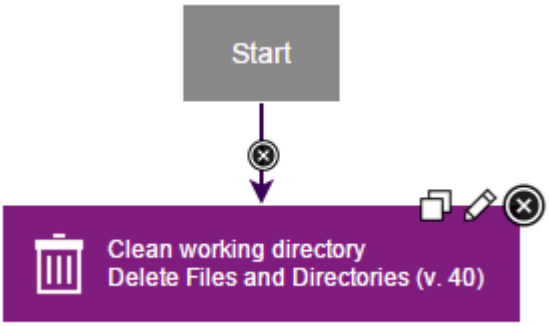
10. **Optional:** If your steps are not connected or are not connected in the correct order, you can connect the steps manually:

    a. If the steps are not in the correct order, remove the connection link between steps by clicking the **X** icons on any of the link arrows; then, rearrange the steps in the correct order.

    b. Move the mouse over the **Start** step, and drag the arrow icon to the `Clean working directory` step.

    Result

    The arrow between these steps represents the flow of activity from one step to another, as in the following figure:



    c. Similarly, drag an arrow from the `Clean working directory` step to the `Download Artifacts` step.

    d. Drag an arrow from the `Download Artifacts` step to the `Remove old content` step.

    e. Drag an arrow from the `Remove old content` step to the `Deploy new content` step.

    f. Drag an arrow from the `Deploy new content` step to the `Finish` step.

Result

You can connect steps in a sequence like these steps, or you can run steps in parallel with each other by creating more links. However, processes always start with the **Start** step and end with the **Finish** step. The sequence of steps in this process is as follows:

a. `Start`
b. `Clean working directory`
c. `Download Artifacts`
d. `Remove old content`

e. `Deploy new content`
f. `Finish`

11. Save the process by clicking **Save**.

Results

The complete component process for the web component looks like the following figure:

Creating a component process for the application component

The application component consists of a single WAR file. This component process downloads the most recent version of the WAR file, updates values in the file, and deploys the file on the application server.

1. From the **Components** tab, click the **JPetStore-APP** component.
2. Click **Processes**, and then click **Create Process**.
3. In the **Create Process** window, specify the name as `Deploy component`.
4. In the **Process Type** list, select **Deployment**.
5. Accept the default values for the other properties, and click **Save**.
6. In the process editor, add a step to clean the working directory.

As you did for the web component, add a **Delete Files and Directories** step named `Clean working directory`, and specify a period in the **Base Directory** field and `**/*` in the **Include** field.

7. Add a step to download the artifacts.
   As you did for the web component, add a `Download Artifacts` step, and accept the default properties.
8. Add a step to extract the WAR file.
   You must extract the WAR file so you can update a property file with the database connection information.

   a. In the list of available steps, expand **Utilities**, and under FileUtils, drag an **Unzip** step to the process editor.

   b. Specify the name as `Expand WAR`.

   c. In the **Extract directory** field, specify the following code:

   ```
   ./JPetStore_expanded
   ```

   d. In the **.zip files** field, specify the following file:

   ```
   JPetStore.war
   ```

   e. In the **Include Files** field, specify the following code:

   ```
   **/*
   ```

   f. Click **OK**.

9. Add a step to update the property file with the location of the database:

   a. In the list of available steps, expand **Utilities**, and from FileUtils add an **Update Java Properties File** step to the process editor.

   b. Specify the name as `Update property file`.

   c. In the **Directory Offset** field, specify the following folder:

   ```
   ./JPetStore_expanded/WEB-INF/classes/properties
   ```

   d. In the **File Includes** field, specify the following file:

   ```
   database.properties
   ```

   e. In the **Add/Update properties** field, specify the following code:

   ```
   url=${p:environment/db.url}
   ```

   This code updates the **url** property in the property file with the location of the database. The `${p:environment/db.url}` code is a variable that represents the location of the database component when the application is deployed.

   f. Click **OK**.

10. Add a step to update the WAR file with the new property file:

a. From the list of available steps, expand the **Utilities**, and from FileUtils, add a **Create .zip File** step to the process editor.

b. Specify the name as `Update WAR`.

c. In the **.zip File Name** field, specify the following file:

```
JPetStore.war
```

d. In the **Base Directory** field, specify the following folder:

```
./JPetStore_expanded
```

e. In the **Include** field, specify the following code:

```
**/*
```

f. Select the **Update Existing** check box.

g. Click **OK**.

11. Add a step to start Tomcat:

a. In the list of available steps, expand **Application Server**, and from Java/Tomcat, add a **Start Tomcat** step to the process editor.
If this step is not in the tree, install the Tomcat plug-in as described in the Prerequisites on page 1935 for this tutorial.

b. Accept the default name `Start Tomcat`.

c. In the **Launcher** field, specify the following code:

```
${p:environment/tomcat.start}
```

This code is another variable that represents the command to start Tomcat.

d. In the **Startup timeout (in seconds)** field, specify `60`.

e. In the **Port** field, specify the port `8081`.

f. In the **Catalina Home** field, specify the home directory where the Apache Tomcat is installed.

g. Click **OK**.

12. Add a step to remove any previous versions of the application:

a. In the list of available steps, expand **Application Server**, and from Java/Tomcat, add an **Undeploy Application** step to the process editor.

b. Accept the default name `Undeploy Application`.

c. In the **Tomcat Manager URL** field, specify the following code:

```
${p:environment/tomcat.manager.url}
```

d. In the **Tomcat Manager User Name** field, specify the user name for the Tomcat server.
You specified this user name and its password in Configuring the agent and target system on page 1935. If you used the code examples from that topic, the user name and password are `tomcatmanager`.

e. In the **Tomcat Manager Password** field, specify the password for the Tomcat server.

f. In the **Context Name** field, specify the following context name, including the preceding forward slash:

```
/${p:environment/tomcat.contextroot}
```

Ensure that you include a slash (`/`) before the variable.

g. Click **OK**.

13. Add a step to deploy the updated WAR file to the application server:

a. In the list of available steps, expand **Application Server**, and from Java/Tomcat, add a **Deploy Application** step to the process editor.

b. Accept the default name `Deploy Application`.

c. In the **Tomcat Manager URL** field, specify the following code:

```
${p:environment/tomcat.manager.url}
```

d. In the **Tomcat Manager User Name** field, specify the user name for the Tomcat server.

e. In the **Tomcat Manager Password** field, specify the password for the Tomcat server.

f. In the **Context Name** field, specify the following context name:

```
/${p:environment/tomcat.contextroot}
```

Ensure that you include a slash (`/`) before the variable.

g. In the **War File Path** field, specify the following path:

```
./JPetStore.war
```

h. Click **OK**.

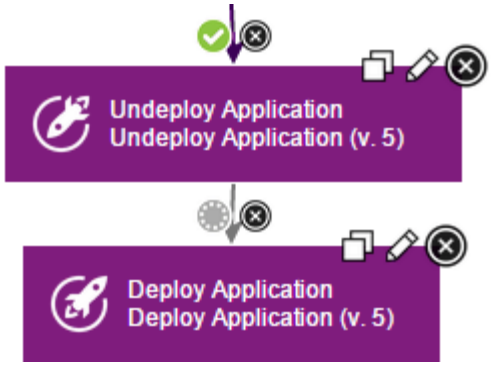
14. **Optional:** If required, arrange and connect the steps in the following order:

a. `Start`
b. `Clean working directory`
c. `Download Artifacts`
d. `Expand WAR`
e. `Update Property File`
f. `Update WAR`
g. `Start Tomcat`
h. `Undeploy Application`

i. `Deploy Application`

j. `Finish`

15. Change the conditional flag on the link between the `Undeploy Application` and `Deploy Application` steps to `both` by clicking the green check mark on the connection until it turns to a gray circle, as shown in the following figure:



Each connection from one step to another has a *conditional flag*. The conditional flag specifies a condition for the connection. Three conditional flags are available:

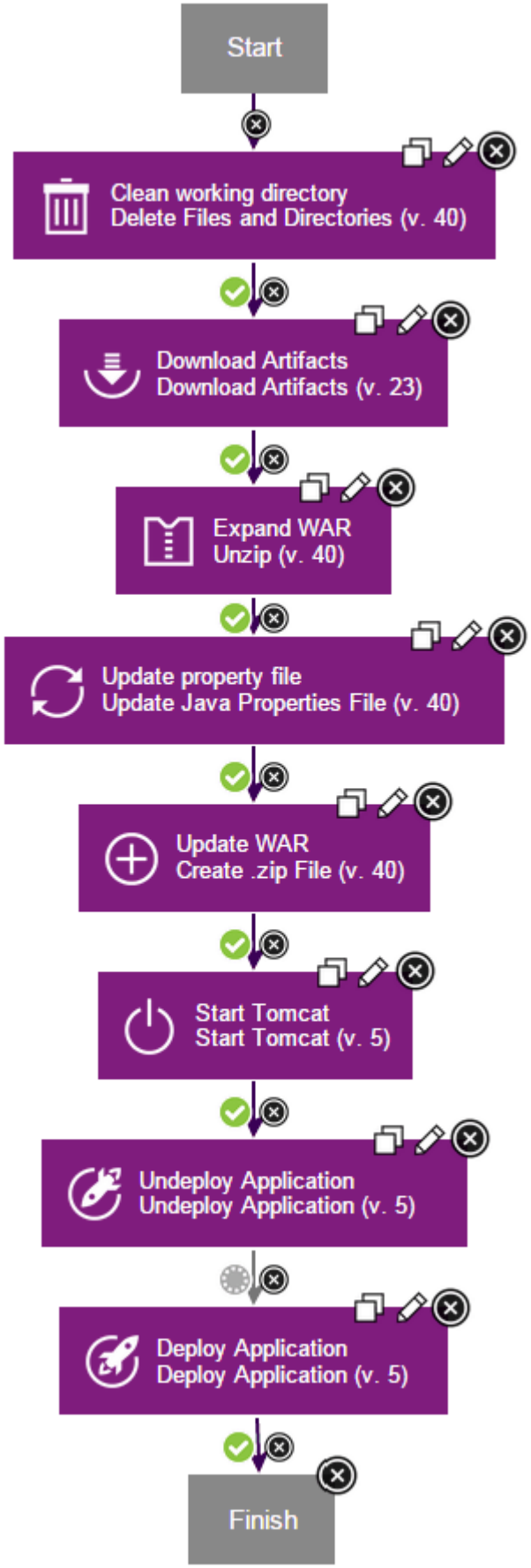
- By default, the conditional flag on each connection is set to `success`, which is represented by a green check mark. The process follows these connections when the step succeeds.
- The conditional flag `fail` is represented by an exclamation point (!) in a red field. The process follows these connections when steps fail.
- The conditional flag `both` is represented by a gray circle. The process follows these connections regardless of whether a step succeeds or fails.

The `Undeploy Application` step fails if no previous versions of the application are installed. In this case, setting the conditional flag on the connection to `both` specifies that the process keeps running regardless of whether the step fails or succeeds.

16. Save the process.

Results

The complete component process for the application component looks like the following figure:


Start
Clean working directory
Delete Files and Directories (v. 40)
Download Artifacts
Download Artifacts (v. 23)
Expand WAR
Unzip (v. 40)
Update property file
Update Java Properties File (v. 40)
Update WAR
Create .zip File (v. 40)
Start Tomcat
Start Tomcat (v. 5)
Undeploy Application
Undeploy Application (v. 5)
Deploy Application
Deploy Application (v. 5)
Finish

Creating a component process for the database component

The deployment process for the database component updates the database with sample values.

1. From the **Components** tab, click the `JPetStore-DB` component.
2. Click **Processes**, and then click **Create Process**.
3. In the **Create Process** window, specify the name as `Deploy component`.
4. In the **Process Type** list, select **Deployment**.
5. Accept the default values for the other properties, and click **Save**.
6. In the process editor, add a step to download the artifacts.
   As you did for the web component, add a `Download Artifacts` step, and accept the default properties.
7. Add a step to update the database:
   You must create a database version table on every target database either manually before you run the DBUpgrader plug-in steps for the first time or as part of the first change.

   a. In the list of available steps, expand **Database**, and under DBUpgrader, add an **Upgrade DB** step to the process editor.
      If this step is not in the tree, install the database upgrade plug-in as described in the Prerequisites on page 1935 for this tutorial.

   b. In the **Edit Properties** window, in the **Name** field, accept the default value of `Upgrade DB`.

   c. In the **Driver Classname** field, specify the following class:

   ```
   com.mysql.cj.jdbc.Driver
   ```

   d. In the **DB Driver Jar** field, specify the following path:

   ```
   lib/mysql-connector-j-8.0.33.jar
   ```

   This file is provided in the source code for the component.

   e. In the **URL** field, specify the following variable:

   ```
   ${p:environment/db.url}
   ```

   f. In the **User** field, specify the user ID of the database user.
      If you used the examples in Configuring the agent and target system on page 1935, the user ID is `jpetstore`.

   g. In the **Password** field, specify the password of the database user.
      If you used the examples in Configuring the agent and target system on page 1935, the password is `jppwd`.

   h. In the **SQL File path** field, specify a single period: `.` .

   i. In the **SQL File Include** field, specify the extension `*.xml` .

   j. In the **Current Version SQL** field, specify the following SQL code:

   ```
   SELECT VER FROM DB_VERSION WHERE RELEASE_NAME = ?
   ```

   k. In the **Delete Version SQL** field, specify the following SQL code:

```
DELETE FROM DB_VERSION WHERE RELEASE_NAME = ?
```

l. In the **Update Version SQL** field, specify the following SQL code:

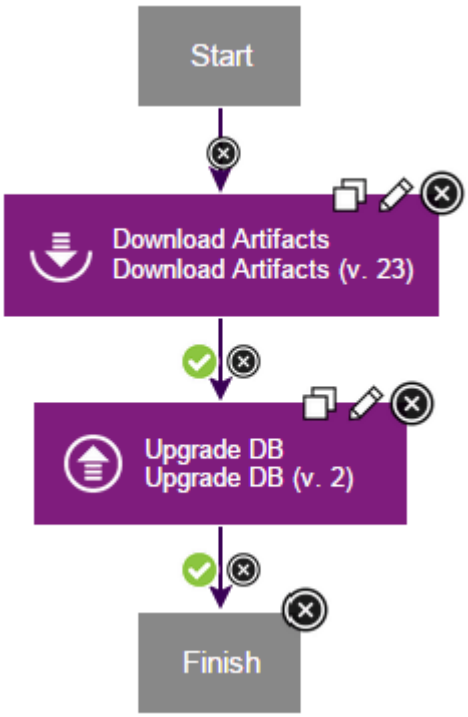
```
INSERT INTO DB_VERSION (RELEASE_NAME,VER) VALUES(?,?)
```

m. Click **OK**.

8. **Optional:** If required, arrange and connect the steps in the following order:
   a. `Start`
   b. `Download Artifacts`
   c. `Upgrade DB`
   d. `Finish`
9. Save the process.

Results

The complete component process for the database component looks like the following figure:



In this lesson, you created component processes to deploy the components.

You can use component processes to automate many different tasks on components. However, you generally do not run component processes directly; in most cases, you add them to application processes. Later in this tutorial, you will create an application process for these three component processes.

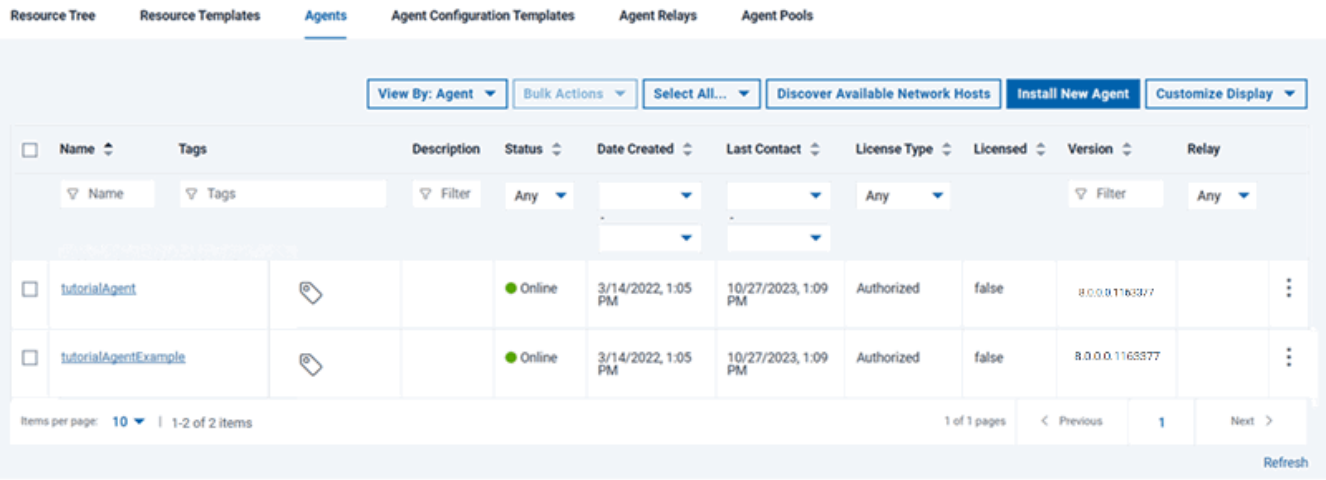
Processes consist of steps, which are provided by plug-ins. To see a list of plug-ins that are available, and for documentation on those plug-ins and steps, see HCL Plug-ins.

Lesson 3: Creating an application and an environment

Applications are collections of components. Environments represent systems on which you deploy the application components.

Before you begin

Before you start this lesson, ensure that your agent is running. Click **Resources > Agents** and make sure that your agent is listed as `Online`, as in the following figure:



About this task

In this lesson, you add the three components to an application and then create an environment on which to deploy the application. You also add your agent to the environment.

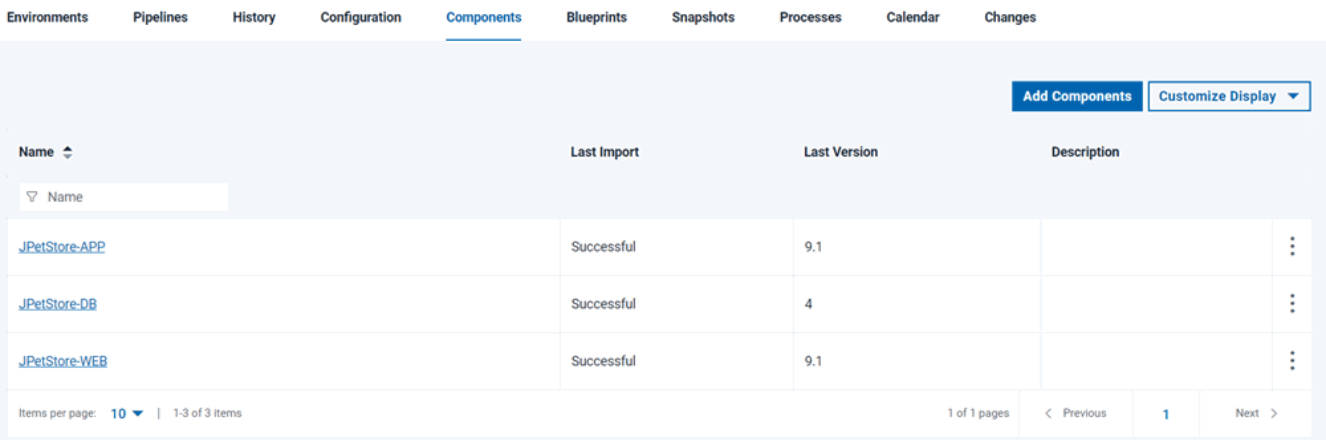
Creating an application

Creating an application involves selecting the components that are in the application.

1. Click the **Applications** tab, and then click **Create Application**.
2. Name the new application `JPetStore`, and then click **Save**.
3. Add the components to the application:
    a. Click the **Applications** tab, and then click the new application, `JPetStore`.
    b. Click the **Components** tab for the application (not the **Components** tab at the top of the page), and then click **Add Component**.
    c. In the **Add a Component** window, in the **Select a Component** list, select the three components: `JPetStore-APP`, `JPetStore-WEB`, and `JPetStore-DB`.
    d. Click **Save**.

Results

The application includes the three components, as shown in the following figure. Components can be included in any number of applications.



Creating an environment

Creating an environment involves selecting one or more agent resources and specifying which components are deployed to each agent resource.

1. Define an environment for the application:
    a. Go to the **Environments** tab for the application.
    b. Click **Create Environment**.
    c. Specify the name to be `Tutorial environment 1`.
    d. Accept the defaults in the other fields in this window and click **Save**.
2. Specify the properties for the environment.
    When you created the component processes, you included parameters that specify the location of artifacts on the target system, such as the location of the application server. Because these properties can vary on different target systems, you specify these properties as environment properties.

    a. Open the environment by clicking the environment name.

    b. Open the environment properties by clicking **Configuration** and then clicking **Environment Properties**.

    c. Add the following properties one at a time by clicking **Add Property** and specifying the name and value of the property as shown in the following table:

    Table 676. Properties for the environment

Property	Description	Example
`tomcat.home`	The Tomcat home folder on the target computer	`/opt/apache-tomcat-9.0.100` on Linux™ or `C:\apache-tomcat-9.0.100` on Windows™
`tomcat.con`	The Tomcat webapp folder on the target computer, which is an identifier in the application URL	For all versions of Tomcat, specify `JPetStoreDev`

Property	Description	Example
`textroot`		
`db.url`	The URL to the MySQL database, relative to the target system	If you installed MySQL on the same system as the agent and used the database name `jpetstore`, the URL looks like the following example: `jdbc:mysql://host-or-ip:3306/jpetstore`
`tomcat.manager.url`	The location of the Tomcat manager application	For Tomcat version 9, specify `http://host-or-ip:8081/manager/text`
`tomcat.start`	The location of the startup script for Tomcat	`/opt/apache-tomcat-9.0.100/bin/startup.sh` on Linux™ or `C:\apache-tomcat-9.0.100\bin\startup.bat` on Windows™

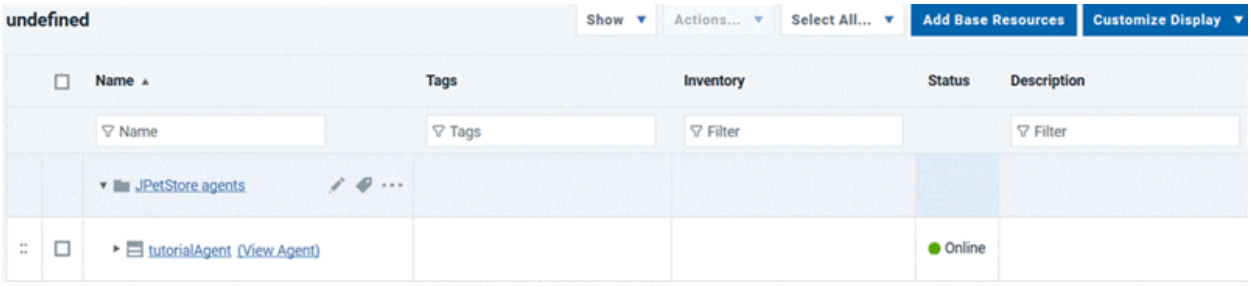
3. Add the agent to the environment.
   When you set up the agent, you added the agent as a resource to a resource group. Now you can use that resource as part of the environment, as described in the following steps:

   a. Click the **Resources** tab for the environment.
      Make sure that you are on the **Resources** tab for the environment and not the **Resources** tab at the top of the page.

   b. Click **Add Base Resources**.
      Result
      The **Add Resource to Environment window shows all of the resources that are available on the server.**

   c. Select the check box next to your agent and then click **OK**.

   Result
   The resource is shown on the **Resources** tab for the environment, as shown in the following figure:



4. Map the components to this agent resource.
   If you had many agents on a system with multiple nodes, you would specify which components run on each agent resource. In this case, you have only one agent resource, so all of the components run on this agent.

a. On the same row as the agent resource, click **Actions > Add Component**.
   The **Actions** button does not appear until you hover the mouse over the resource.

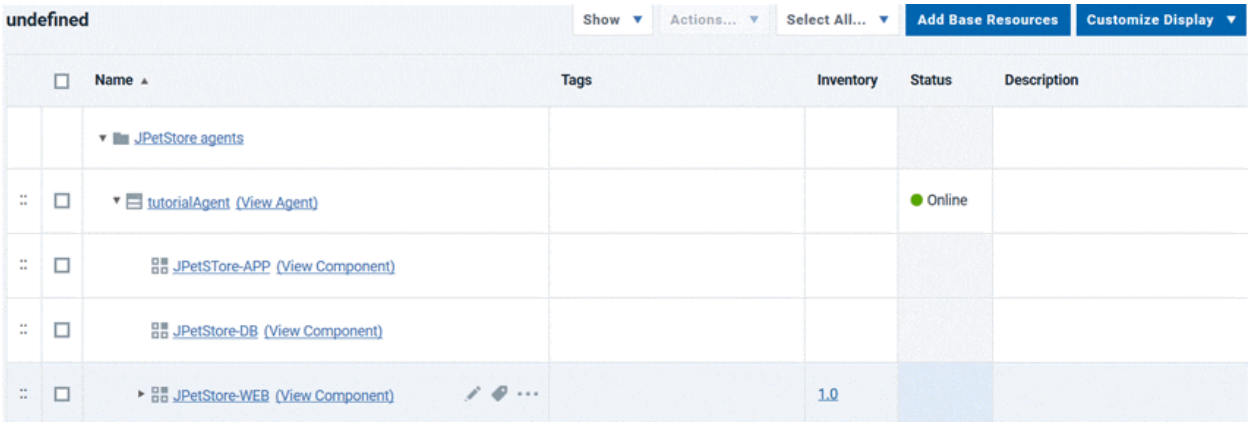
b. Select the application component and then click **Save**.

c. Repeat the process to add the other components to the resource.

Note: Be sure to add each component to the agent resource and not to another component. The components must be at the same hierarchy level, and not as child items of each other.

Result

Now the components are mapped to the target system, as shown in the following figure:



Results

The new environment appears on the **Environments** tab. Now that the environment is prepared and the agent is running, you can run application processes on the environment to deploy the components.

Lesson 4: Creating an application process

Application processes tie together component processes.

About this task

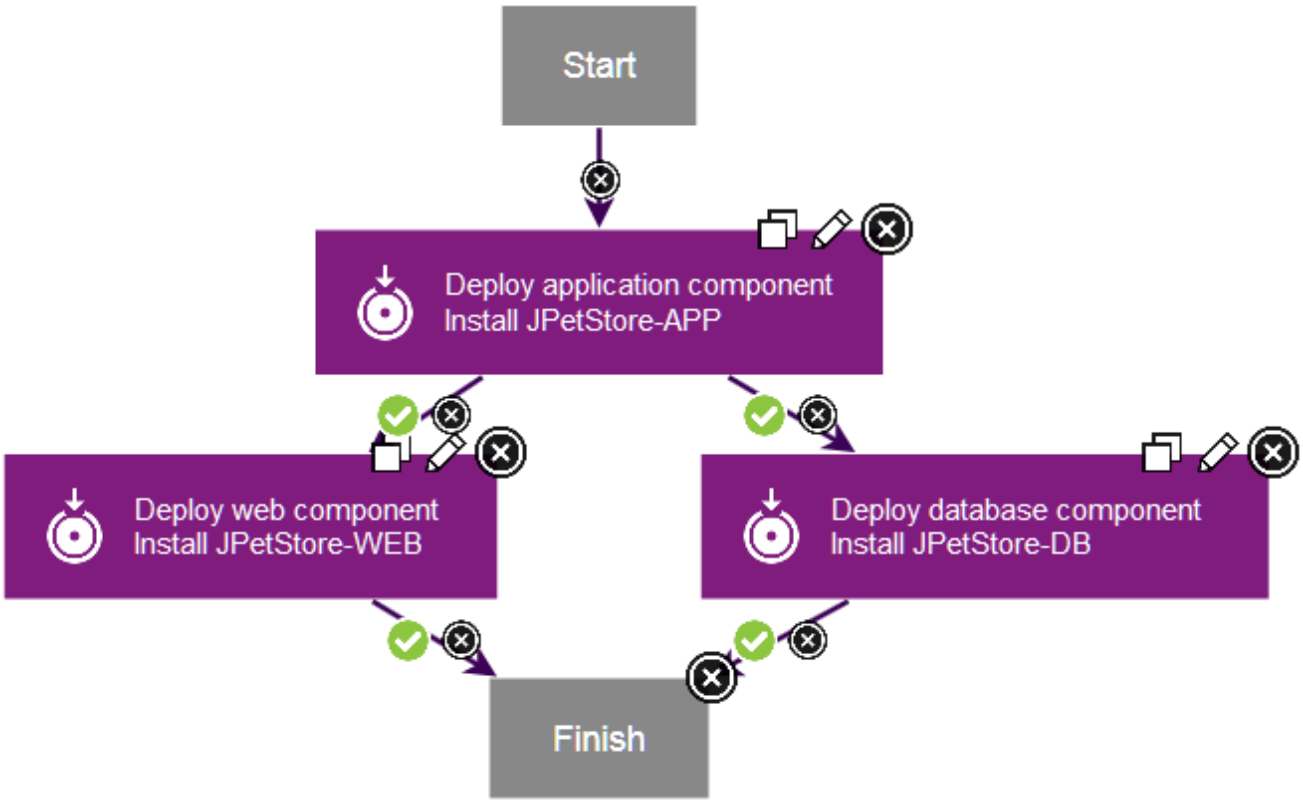
In this lesson, you create an application process to install the components by calling each component process.

1. Click the **Applications** tab, and then click the **JPetStore** application.
2. Click **Processes**, and then click **Create Process**.
   Be sure to click the **Processes** tab within the application and not the **Processes** tab at the top of the page, which leads to generic processes.
3. In the **Create an Application Process** window, name the new application process `Deploy JPetStore`.
4. Accept the default values for the other fields, and click **Save**.
5. Click the new process to open it in the process editor.
   The process editor for application processes is similar to the process editor for component processes. However, for application processes, the steps are limited to calling component processes and a few utility steps.

6. Add a step to deploy the application component:
    a. From the list of steps, add an **Install Component** step to the process editor.
    This step calls component processes that have a process type of `Deployment`.
    b. In the **Edit Properties** window, name the step `Deploy application component`.
    c. In the **Component** list, select the `JPetStore-APP` component.
    d. In the **Component Process** list, select the `Deploy Component` component process.
    e. Click **OK**.
7. Similarly, add a step that is named `Deploy component` to deploy the web and database components.
8. Verify that the `Start` step is connected to the `Deploy application component` step.
9. Instead of connecting the remaining steps in a straight line, follow these steps to connect the steps so that they run at the same time.
    Use the figure at the end of this step as a reference point.
    a. Connect the `Deploy application component` step to the `Deploy web component` step.
    b. Connect the `Deploy application component` step to the `Deploy database component` step.
    Result
    Now these two steps will run at the same time, after the application step.
    c. Connect both the `Deploy web component` step and the `Deploy database component` step to the `Finish` step.
10. Save the process.

Results

The complete application process deploys the application component first and then deploys the web component and database components at the same time. The application process looks like the following figure:



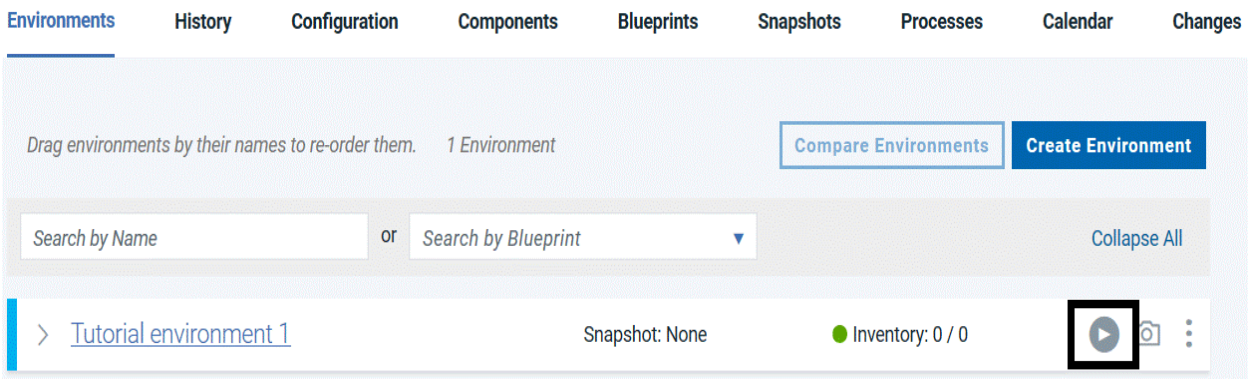
Lesson 5: Deploying the application components

To deploy the components in the application, run the application process on the new environment.

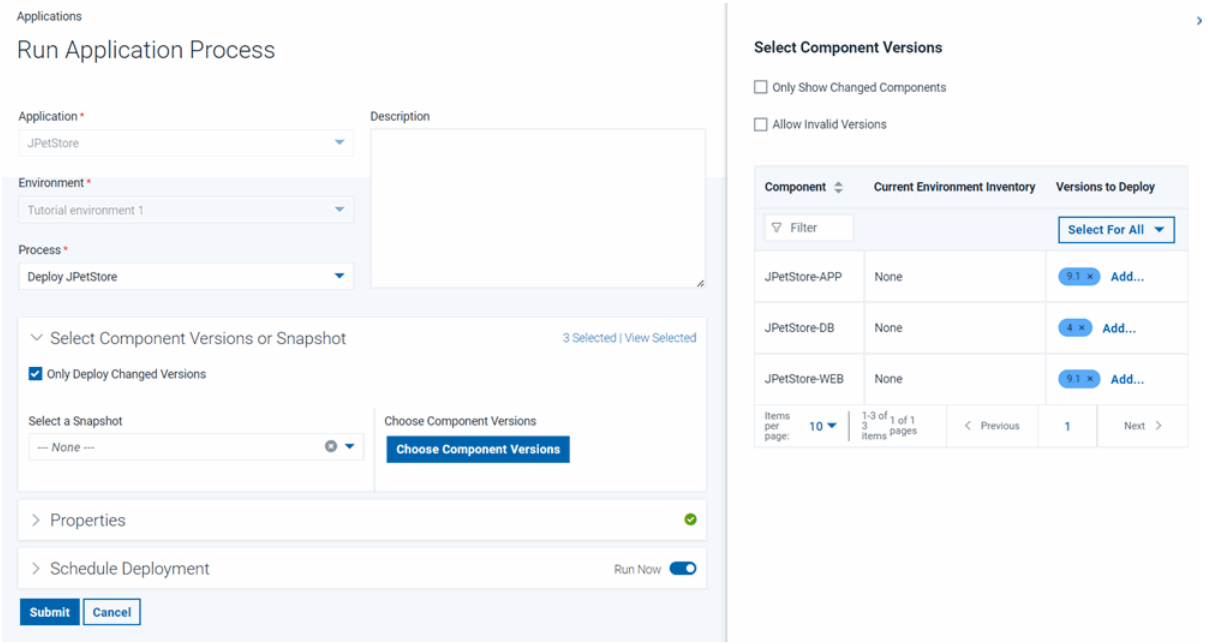
About this task

Now that you have an environment and all the necessary processes, you can deploy the components by running the application process.

1. Open the application page by clicking **Applications** and then clicking the application name.
2. In the same row as your environment, click the **Request Process** icon, as shown in the following figure.



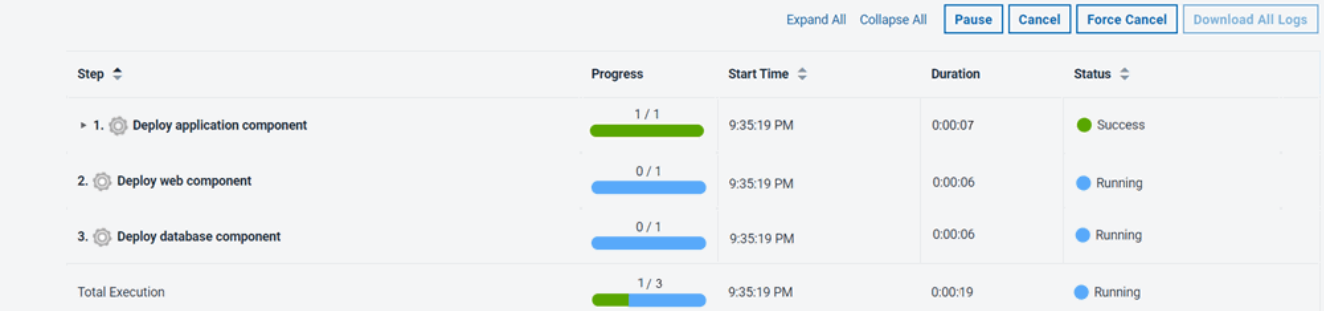
3. In the **Process** field, select the `Deploy JPetStore` process.
4. Go to **Choose Component Versions** and click **Select For All > Latest Available**.
   Make sure that version 1.0 is selected for each component, as shown in the following figure:



   You must select a version for each component; if you do not select a version, that component is not deployed.
5. Review the deployment settings and click **Submit**.

Results

The web page shows you the progress of the application process request. From this page, you can watch as the processes run. The following figure the application process partially completed. The application component process is finished and the other two component processes are running.



If the process runs successfully, the request shows that each component process is finished, as in the following figure:

Step	Progress	Start Time	Duration	Status
1. Deploy application component	1 / 1	12:28:47 PM	0:01:00	Success
1. Deploy web component	1 / 1	12:28:48 PM	0:00:15	Success
1. Deploy database component	1 / 1	12:28:49 PM	0:00:15	Success
Total Execution	3 / 3	12:28:47 PM	0:01:30	Success

In this case, you can view the running application at the following URL, by substituting the host name of your target system for *hostname*. This is the host name of the agent that you configured in Configuring the agent and target system on page 1935. In the URL, specify the *tomcat.contextroot* value that you entered for this environment after the port number.

```
http://hostname:8081/JPetStoreDev
```

The application shows a simple online shopping site in the form of a pet store. The following figure shows the home page:

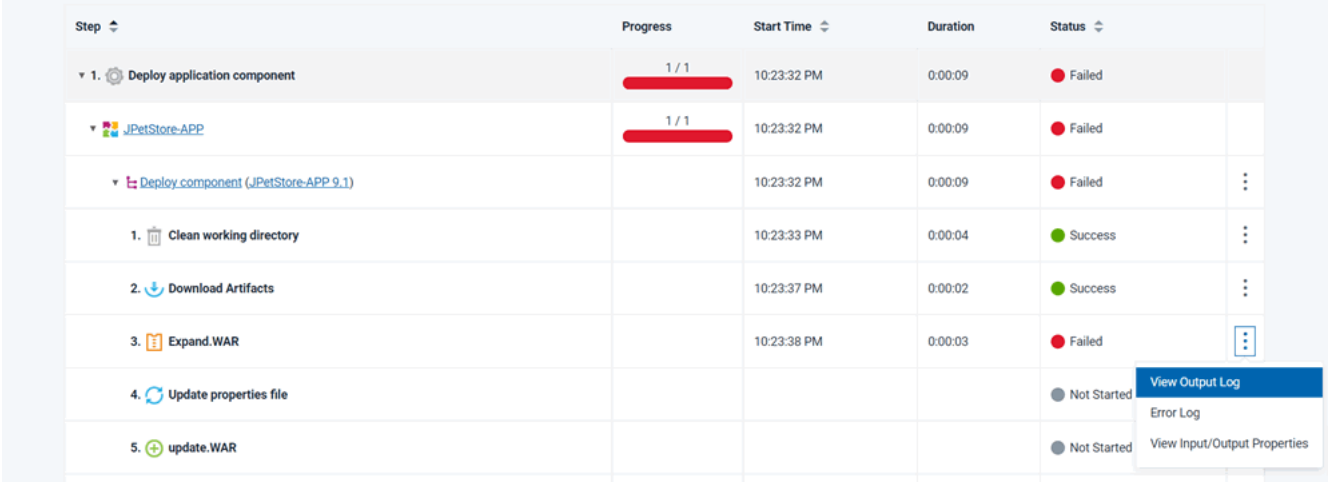
Welcome to JPetStore 4

Enter the Store

Powered By
iBATIS
© Clinton Begin

From the home page, click **Enter the Store** and then browse the items that are available, as shown in the following figure:

If the application process did not run successfully, one of the component processes might have failed. In this case, the process is listed with a status of `Failed`. To troubleshoot this failed process, expand the log section for the component, and find the step that failed:



Click **View Output Log** to look at the command-line output log for each step in the process..

If one of your component processes did not complete, identify the step that did not complete. Verify that the properties for that step are correct and that the steps are in the correct order in the process. Then, run the application process again.

Note: If you run the application process again, be sure to clear the **Only Changed Versions** check box and select the versions of each component. If you leave this check box selected, the server runs the component processes only for components that have new versions.

This application process installed each of the components in the application by running their component processes.

Application processes can also uninstall or update components or run other configuration tasks for the application and its components.

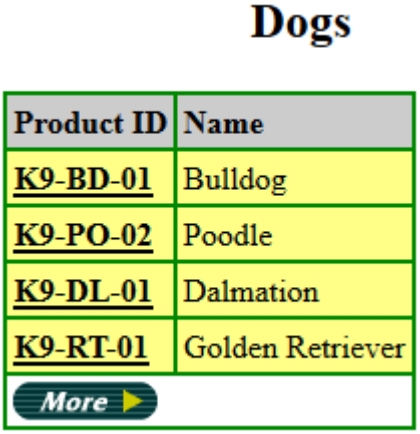
Lesson 6: Updating the application

In the first lesson, you deleted certain versions of the components. In this lesson, you add those versions again to simulate deploying a new version of the components.

About this task

The web application that you deployed in the previous lesson included version 1 of the web component and database component. Version 1.1 of these components includes a new item for the online store: a Bichon in the Dogs category.

1. Verify that `Bichon` is not available in the Dogs category:
    a. Open the application by opening a web browser to the following URL:
    ```
    http://hostname:8081/JPetStoreDev
    ```
    b. Click **Enter the Store**.
    c. Browse the Dogs category and verify that `Bichon` is not available.
    The following figure shows some of the dogs in this category:



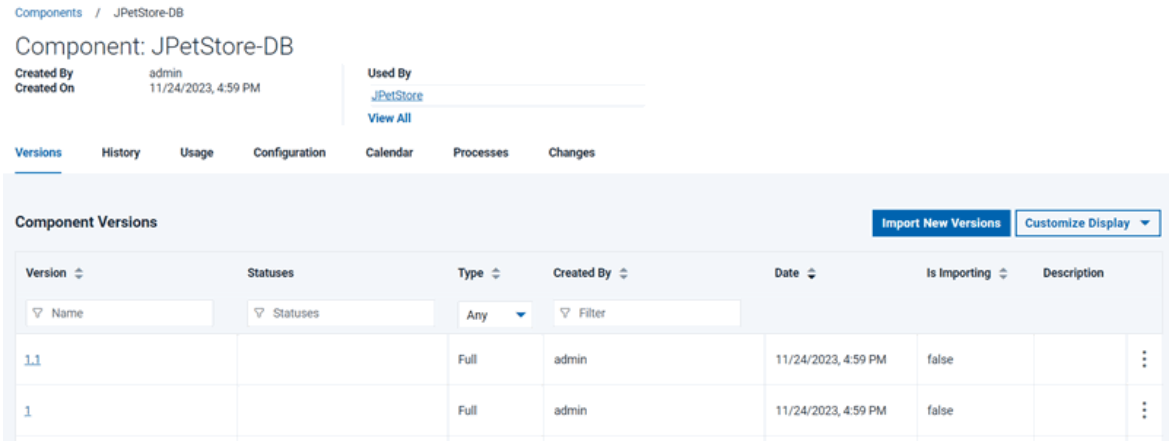
2. Import the new version of the database component:

a. On the Deploy server, click the **Components** tab and then click the JPetStore-DB component.

b. Click the **Versions** tab and then click **Import New Versions**.

Result

Now, version 1.1 is shown in the list of versions, as shown in the following figure:



c. Click the new version of the database component to open its information page.

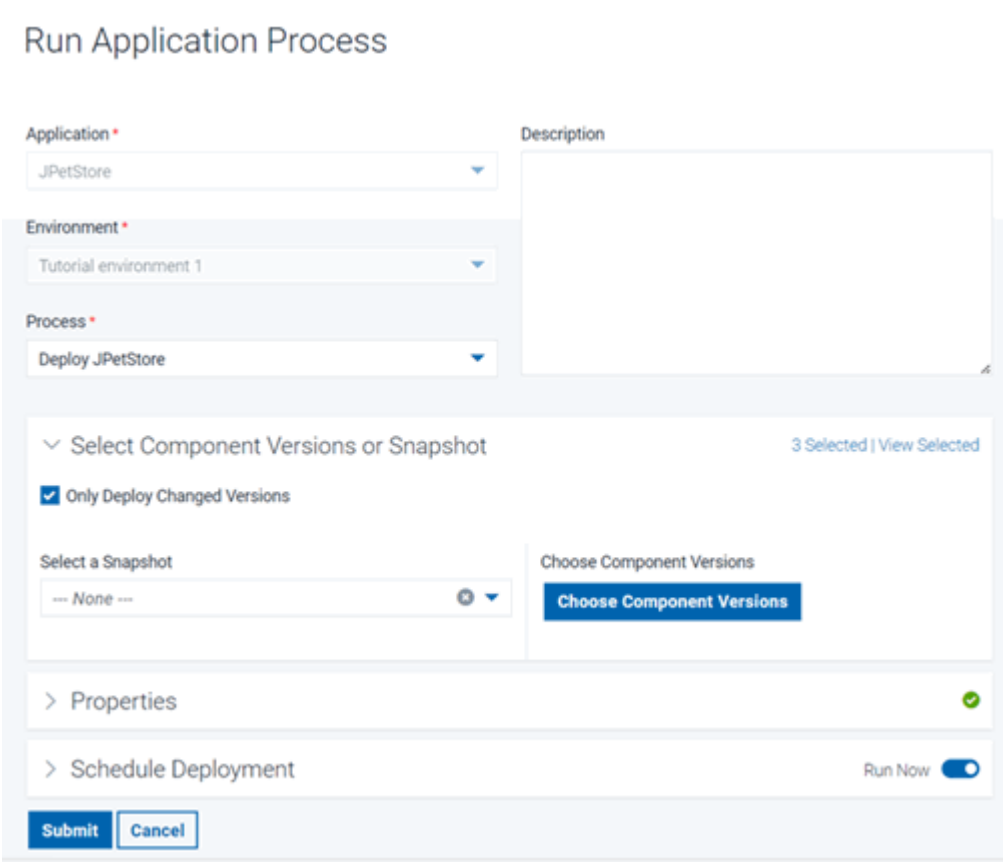
d. Click the **Configuration** tab.

e. Under **Basic Settings**, in the **Type** list, select **Incremental** and then click **Save**.

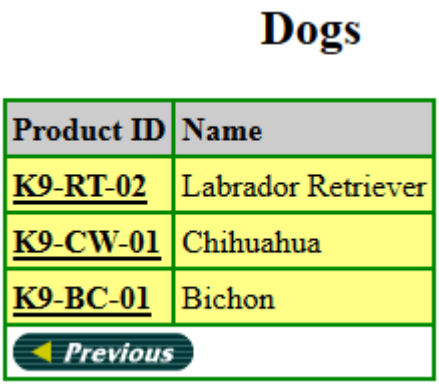
Result

This version is set to be an incremental upgrade rather than a new version. Full versions are installed directly, without installing prior versions first. Incremental upgrades start with prior versions, and then the new version is installed over the prior versions. The new database version is only an addition to the database, so it requires the prior version of the component.

3. In the same way, import version 1.1 of the web component without changing the **Type** setting.
   The application component does not have any additional versions.
4. Click **Applications** and then click the JPetStore application to get back to the list of environments.
5. On the environment, run the application process again.
   In the **Run Process** window, be sure to select the latest version for each of the components by clicking **Choose Versions** and then clicking **Select For All > Latest Available**. Optionally, you can select the **Only Changed Versions** check box so that the server deploys only the components with new versions. In this case, the server runs the component processes only for the web and database components.



6. In your web browser, reload the web application and explore the Dogs category to see the new `Bichon`, as shown in the following figure:



   This new item demonstrates that the database component is updated to version 1.1
7. Click the Product ID for `Bichon` and then the Item ID to see the product page for `Bichon`.

The new graphic for this item demonstrates that the web component is updated to version 1.1.

Results

Now the application includes the new versions of the components.

In a production scenario, new versions of components might become available frequently. You can run the application process as often as necessary to update the deployed components. You can also configure the application process to run automatically when new versions of the components are available.

Now that you have completed this tutorial, you can continue learning about how to deploy components by using snapshots. See the tutorial Creating and deploying snapshots on page 1970.

Summary

In this tutorial, you learned to deploy and update an application with HCL DevOps Deploy (Deploy).

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How Deploy structures applications and manages automated deployment instructions
- How to import the components of an application
- How to create processes that specify automated tasks for components and applications
- How to configure environments for deployments
- How to manage the component versions on an environment by running processes

Creating and deploying snapshots

In this tutorial, you deploy a simple web application to a new environment with HCL DevOps Deploy (Deploy). You create a snapshot of the components in an application, and then deploy the snapshot to an environment.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a snapshot of an environment's components to quickly deploy its application to another environment.
- Configure environments on which to deploy applications.
- Select a snapshot while you run an application process to promote the snapshots to a new environment.

Time required

Approximately 1 hour

Note: You must complete Deploying a simple web application on page 1934 before you begin this tutorial.

Introduction

In this tutorial, you deploy a simple web application to a new environment with HCL DevOps Deploy (Deploy). You create a snapshot of the components in an application, and then deploy the snapshot to an environment.

You can use Deploy to facilitate a continuous deployment pipeline in your release cycle. You can plan and automate the deployment of complex applications to development, test, and production environments. You can use this tool to deploy applications as often as needed and to as many different environments as you require. By using snapshots, you combine the planning and automation capabilities of Deploy with the ability to quickly and precisely control the contents of your delivery pipeline's environments.

When you develop a complex application, you use many tests to validate that all the application components work together. After your application passes the tests, you must ensure that you promote the correct collection of application components to the next step in your continuous delivery pipeline. However, you must ensure that you promote the set of components that passed the tests. In Deploy, you use snapshots. A snapshot specifies all the versions of an application’s components that are used in an environment. The snapshot also specifies versions of configuration, such as deployment processes, that are used to deploy those components. If you use snapshots to deploy to different environments in your continuous delivery pipeline, you no longer must track and promote individual components through the environments in your pipeline. Instead, you promote configurations of components that were tested together. This ability to promote tested snapshots of components to environments is essential in maintaining a continuous delivery pipeline.

In this tutorial, you use snapshots to deploy the components, application, and processes that you created for a three-tiered web application to a new environment. In most cases, you use snapshots to manage complex deployment environments in Deploy. You typically use snapshots to control deployments to different environments on different servers. However, for simplicity, in this lesson you use snapshots to deploy an application to two different environments on a single server.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Create a snapshot of an environment's components to quickly deploy its application to another environment.
- Configure environments on which to deploy applications.
- Select a snapshot while you run an application process to promote the snapshots to a new environment.

Time required

This tutorial takes approximately 1 hour to finish. If you explore other concepts that are related to this tutorial, it might take longer to complete.

System requirements

To run this tutorial, you must have an installation of Deploy, version 6.0 or later

Prerequisites

- Make sure that you completed the Deploying a simple web application on page 1934 tutorial and have access to its Deploy server and computers, virtual images, or cloud instances.
- You must have an account on the server with the permissions to create and edit environments and to run processes.
- You must know the user names and passwords for the MySQL database and Apache Tomcat manager role that you configured in the Deploying a simple web application on page 1934 tutorial.

Configuring the server and target system

Modify the HCL DevOps Deploy (Deploy) server that you used in the previous tutorial. Create another agent resource and incorporate a new environment property to allow for easy reuse.

Before you begin

Before you begin the tutorial, make sure that you completed the Prerequisites on page 1971.

About this task

To work through this tutorial, you must use the target system that you configured in the Deploying a simple web application on page 1934 tutorial.

This system hosts the following items that are associated with your Deploy JPetStore application:

- A Deploy agent
- A MySQL database
- An instance of Apache Tomcat

You must have MySQL and Apache Tomcat installed on the system. Follow these steps to set up an agent resource, create a new MySQL database, and redeploy your application to your environment:

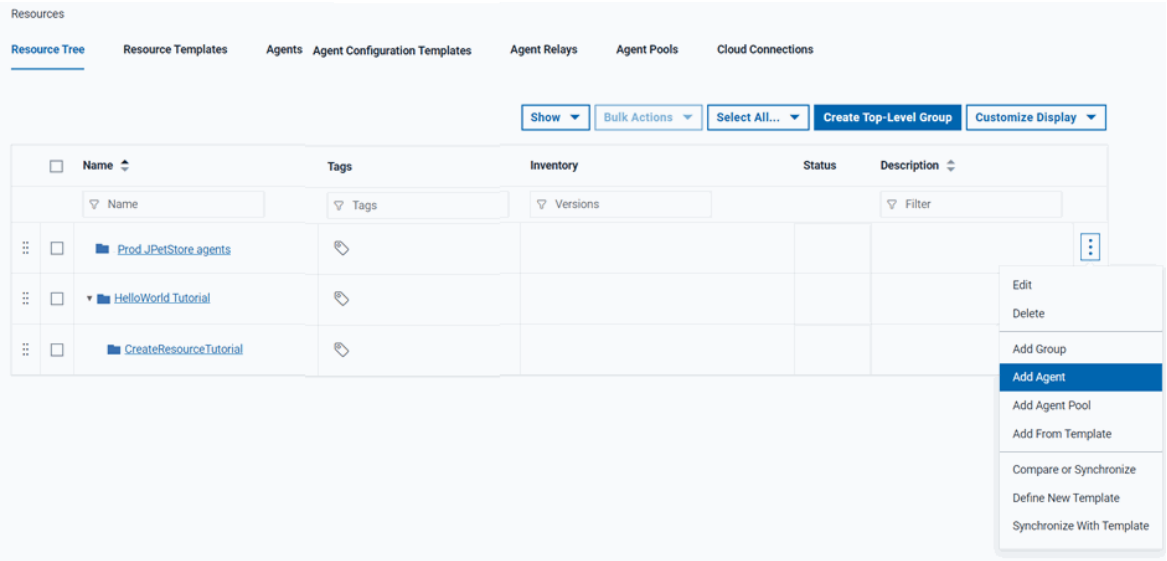
1. Create another resource and add the existing agent to it.
   To reuse the agent with another environment on the same Deploy server, you must associate the agent with another resource.

   a. On the Deploy server, click the **Resources** tab.
      Be sure to click the **Resources** tab at the top of the page.

b. From the **Resource Tree** tab that is displayed, click **Create Top-Level Group**.

c. In the **Create Resource** window, provide the new group with a name such as `Prod JPetStore agents` and click **Save**.

d. On the same row as the new resource group, click **Actions > Add Agent**, as shown in the following figure:



e. In the **Create Resource** window, select the agent in the **Agent** field and click **Save**.

Result

The agent is listed as a child of both resource groups. By associating the agent with both resource groups, you can use the same agent with your existing environment and the one that you create in a later lesson.

2. Create an empty database for the application to use in another environment:

a. Log in to the MySQL command line as the MySQL root user.

Example

For example, if you installed MySQL as the root user, run this command from the command line:

```
mysql -u root
```

Result

The command line shows a prompt that looks like this prompt:

```
mysql>
```

b. Create the database with the following command:

```
create database jpetstoreprod;
```

Result

If the command runs correctly, you see this message: `Query OK, 1 row affected`.

c. Give the existing database user access rights to the database.

The database user that has access to the `jpetstoredev` database must have access to the new database too. If you used the default user name in the Deploying a simple web application on page 1934 tutorial, create another database by using the following command:

```
grant all privileges on jpetstoreprod.* to 'jpetstore'@'host-or-ip';
```

Note: If you used the default settings when you configured the original database, the password for user `jpetstore` is `jppwd`.

Result

If the command runs correctly, you see this message: `Query OK, 0 rows affected`.

d. Type `quit` to exit the MySQL command line.

3. Verify that the user can access the database:

a. From the command line, log in to MySQL as the existing user:

```
mysql -u jpetstore -pjppwd
```

b. Run the following command to list the databases on the system:

```
show databases;
```

c. Verify that both the `jpetstoreprod` and `jpetstoredev` databases appear in the list, as shown in the following figure:

Result

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jpetstoredev       |
| jpetstoreprod      |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

d. Type `quit` to exit the MySQL command line.

4. Before you deploy your application, confirm that all environment properties contain the correct values:

a. From the **Applications** tab of your Deploy server, click the `JPetStore` application.

b. Open the environment by clicking the environment name.

c. Open the environment properties by clicking **Configuration** and then clicking **Environment Properties**. Be sure to click the **Configuration** tab for your application, not the **Configuration** tab at the top of the page.

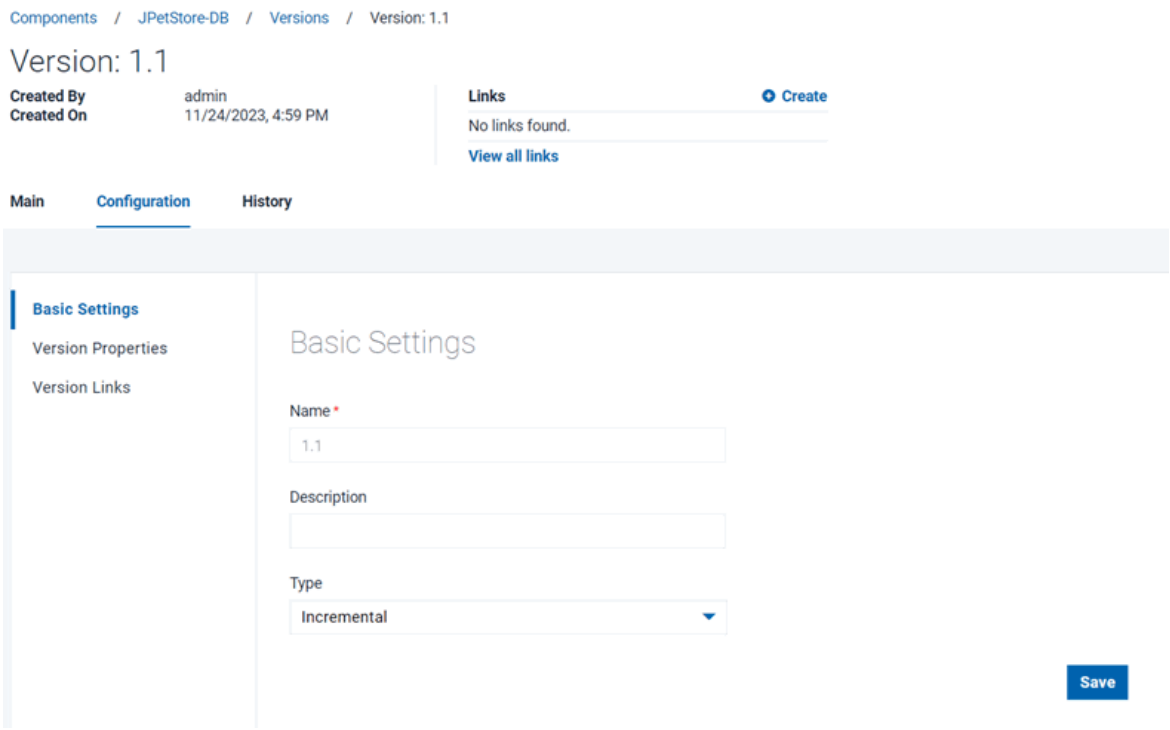
Result

The environment properties are displayed as shown in the following table:

Table 677. Properties for the environment

Property	Description	Example Value
`tomcat.home`	The Tomcat home folder on the target computer	`/opt/apache-tomcat-9.0.100` on Linux™ or `C:\apache-tomcat-9.0.100` on Windows™
`tomcat.contextroot`	The Tomcat webapp folder on the target computer, which is an identifier in the application URL	For Tomcat version 9, specify `JPetStoreDev`
`db.url`	The URL to the MySQL database, relative to the target system	If you installed MySQL on the same system as the agent and used the database name `jpetstore`, the URL looks like the following example: `jdbc:mysql://host-or-ip:3306/jpetstore`
`tomcat.manager.url`	The location of the Tomcat manager application	For Tomcat version 9, specify `http://host-or-ip:8081/manager/text`
`tomcat.start`	The location of the startup script for Tomcat	`/opt/apache-tomcat-9.0.100/bin/startup.sh` on Linux™ or `C:\apache-tomcat-9.0.100\bin\startup.bat` on Windows™

5. To ensure that the `JPetStore-DB` component is deployed correctly, confirm that component version `1.1` is an incremental version.
   An incremental component version contains only the changes that are associated with that version. For the component to be complete, you must deploy the previous versions of the component as well.
   a. From the **Components** tab, click the `JPetStore-DB` component.
   b. Click the **Versions** tab to display the available component versions, and then click `1.1` to display its attributes.
   c. Click the **Configuration** tab. Make sure to click the **Configuration** tab that is within the component, not the **Configuration** tab at the top of the page.
   d. On the **Basic Settings** panel, confirm that `Incremental` is selected from the **Type** list, as shown in the figure:



e. If `JPetStore-DB` version `1.1` is not set to **Incremental**, from the **Type** list, select **Incremental**, and then click **Save**.

6. To confirm that you configured your environment and component correctly, redeploy the application to your environment:

   a. Click the **Applications** tab, and then click the `JPetStore` application.

   b. In the same row as your environment, click the **Request Process** icon.
   Result
   The **Run Process** window opens.

   c. Clear the **Only Changed Versions** check box.

   d. In the **Process** list, select `Deploy JPetStore`.

   e. Click **Choose Versions**.
   The **Component Versions** window opens.

   f. In the **Select For All** list, click **Latest Available**.

   g. In the same row as `JPetStore-DB`, click **Add**.

   h. From the displayed list, select `1.0`.
   Result
   Your list of **Versions to Deploy** is displayed.

i. To record your version selections, click **OK**.

j. To deploy your application, click **Submit**.

Result

Deploy runs the application process, and your application is deployed.

Results

The agent and the target system are configured for your work with the sample application. You need the password that you used and locations that you specified in this lesson later.

Lesson 1: Adding snapshots

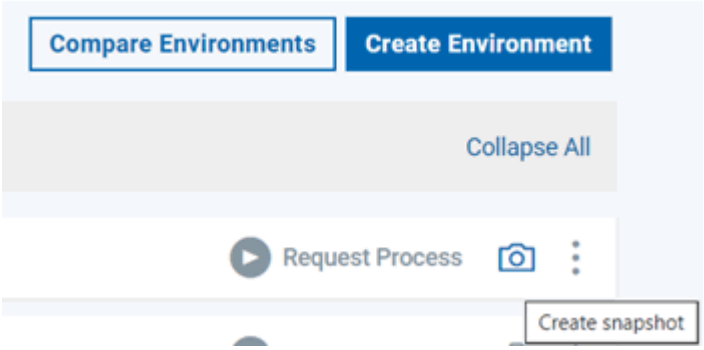
Creating a snapshot involves saving a collection of component versions that are part of an application.

About this task

These steps describe how to create a snapshot that uses components of the sample application. In the Deploying a simple web application on page 1934 tutorial, you deployed the JPetStore application twice to the same environment. In the second deployment, you used two components that contain new information. In this lesson, you create a snapshot that uses the same components as the deployed environment. You create a snapshot from a functioning development environment in preparation for deployment to the next phase of your continuous delivery pipeline. In a later lesson, you deploy this snapshot to a new environment. By deploying with snapshots in Deploy, you can quickly bring a new environment to a specific state.

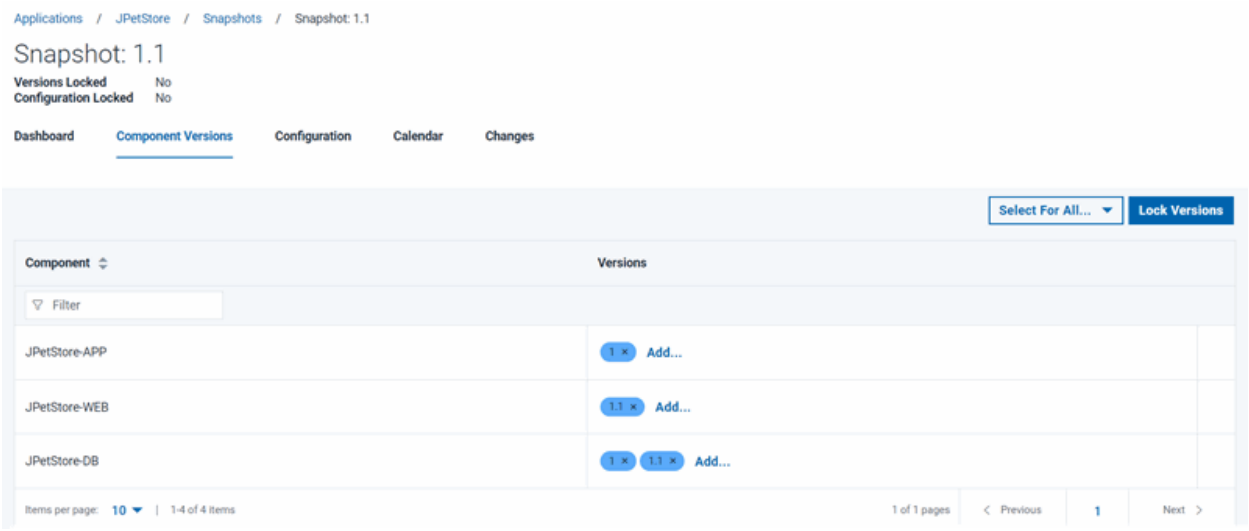
Before you begin: Make sure that your Deploy server has the prerequisites that are listed in Prerequisites on page 1971 and that you completed the steps in Configuring the server and target system on page 1971.

1. Click the **Applications** tab, and then click `JPetStore`.
2. In the same row as your environment, click the **Create Snapshot** icon, as shown in the following figure:



3. In the **Create Snapshot** window, specify the name to be `Version 1.1`.
4. Accept the default values in the other field, and then click **Save**.
5. Click the **Component Versions** tab.

   The snapshot is shown as in the following image.



The `JPetStore-DB` component uses incremental versions and must contain both the `1.0` and `1.1` versions to deploy. If the `JPetStore-DB` component does not contain both versions, you must update the component version. See Configuring the server and target system on page 1971.

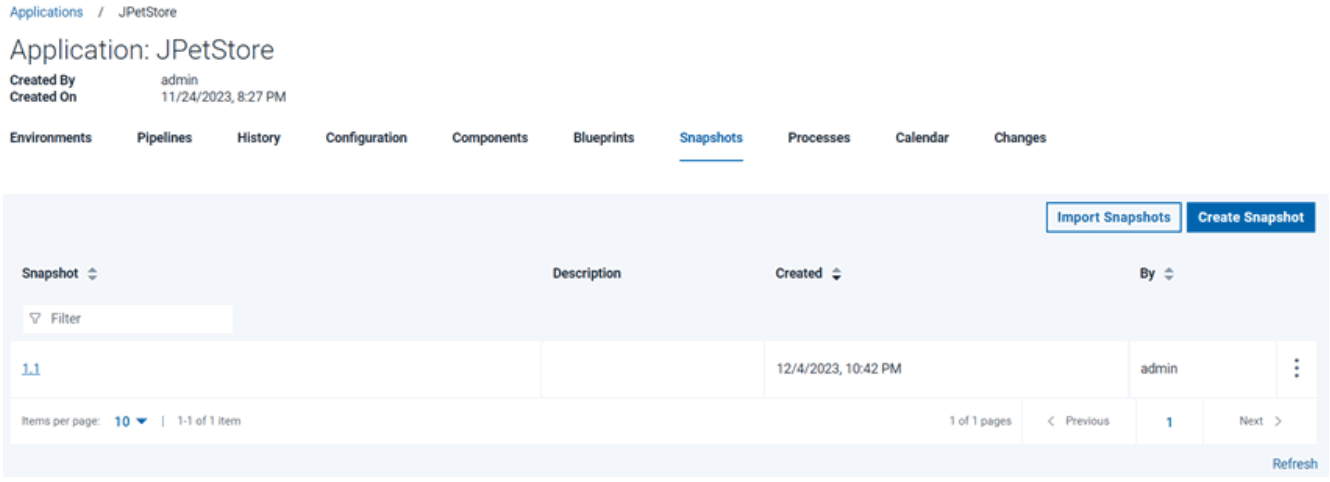
Result

The `JPetStore-DB` component uses incremental versions and must contain both the `1.0` and `1.1` versions to deploy. If the `JPetStore-DB` component does not contain both versions, you must update the component version. See Configuring the server and target system on page 1974.

Results

In this lesson, you added a snapshot to the application. The snapshot is based on the components in the application that are currently deployed to an environment.

On the application's **Snapshots** page, you have a snapshot, as shown in the following figure:



In the next lesson, you create an environment to for the next step in the JPetStore application's delivery pipeline.

What to do next

After you create a snapshot that describes a collection of components, you can deploy the snapshots to an environment. In the next lesson, you create another environment.

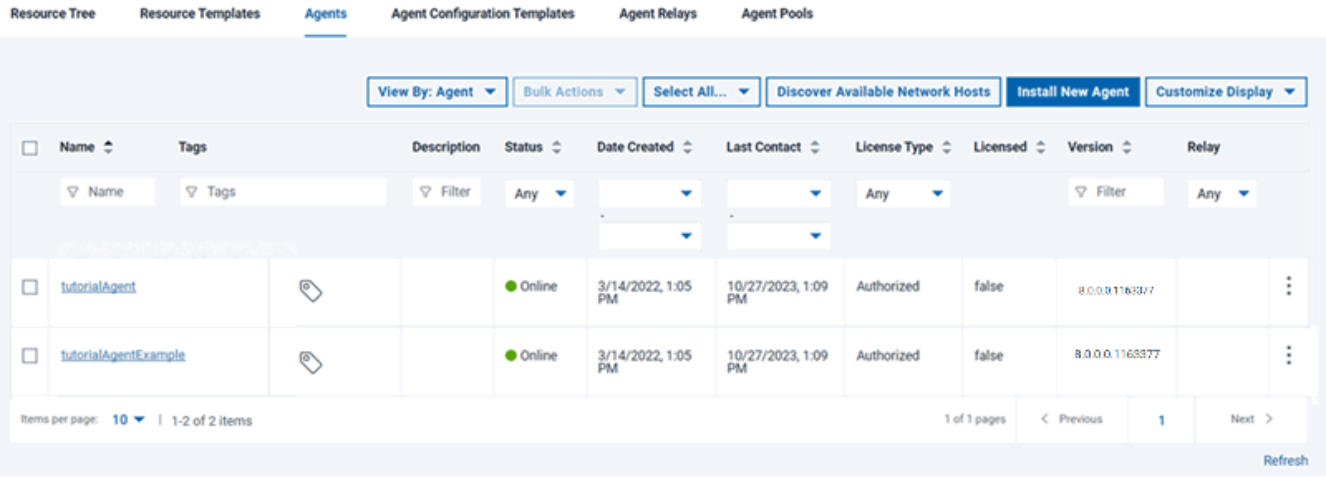
For more information about creating snapshots, see Creating snapshots on page 2383.

Lesson 2: Creating an environment

Environments are deployment targets for the application components that you specify. You create another environment in the application's delivery pipeline to facilitate multiple active applications. You apply an agent resource and specify which components the agent resource deploys.

Before you begin

Before you start this lesson, ensure that the agent is running. Click **Resources > Agents** and make sure that your agent is listed as `Online`, as in the following figure:



About this task

In this lesson, you create an environment to deploy the snapshots that you created. You also add your agent to the environment. By adding a second environment, you can mimic a simple delivery pipeline that has two environments.

1. Define an environment for the application:
    a. Click the **Applications** tab, and then click `JPetStore`.
    b. From the **Environments** tab for the application, click **Create Environment**.
    c. Specify the name to be `Tutorial environment 2`.
    d. Accept the default values in the other fields in this window, and click **Save**.
2. Specify the properties for the environment.
    When you created the component processes, you included parameters that specify the location of artifacts on the target computer system, such as the location of the application server. Because these properties can vary on different target systems, you specify these properties as environment properties.

    a. Open the environment by clicking the environment name.

    b. Open the environment properties by clicking **Configuration** and then clicking **Environment Properties**.

c. Add the following properties one at a time by clicking **Add Property** and specifying the name and value of the property as shown in the following table:

Table 678. Properties for the environment

Property	Description	Example Values
`tomcat.home`	The Tomcat home folder on the target computer	`/opt/apache-tomcat-9.0.100` on Linux™ or `C:\apache-tomcat-9.0.100` on Windows™
`tomcat.contextroot`	The Tomcat webapp folder on the target computer, which is an identifier in the application URL	For Tomcat version 9, specify `JPetStoreProd`
`db.url`	The URL to the MySQL database, relative to the target system	If you installed MySQL on the same system as the agent and used the database name `jpetstoreprod`, the URL looks like the following example: `jdbc:mysql://host-or-ip:3306/jpetstoreprod`
`tomcat.manager.url`	The location of the Tomcat manager application	For Tomcat version 9, specify `http://host-or-ip:8081/manager/text`
`tomcat.start`	The location of the start script for Tomcat	`/opt/apache-tomcat-9.0.100/bin/startup.sh` on Linux™ or `C:\apache-tomcat-9.0.100\bin\startup.bat` on Windows™

3. Add the agent to the environment.
   You added the agent as a resource to another resource group. Next, you add that resource to the environment, as described in the following steps:

   a. Click the **Resources** tab for the environment.
      Make sure that you are on the **Resources** tab for the environment and not the **Resources** tab at the top of the page.

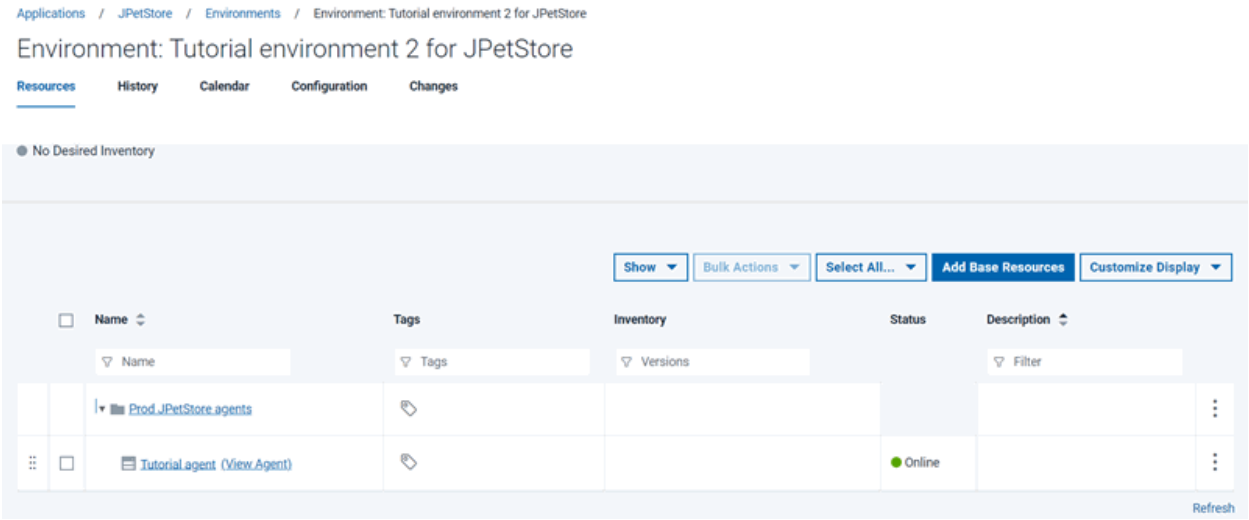
   b. Click **Add Base Resources**.
      Result
      The **Add Resource to Environment** window shows all the resources that are available on the server.

c. Select the check box next to your agent in the `Prod JPetStore agents` resource, and then click **OK**.

Note: Since one agent deploys the components for in both application environments, ensure that the agent you select is in the agent resource that you created for your new environment. You must not select the agent in the agent resource that is associated with the application's other environment.

Result

The resource is shown on the **Resources** tab for the environment, as shown in the following figure:



4. Map the components to this agent resource.
   If you had many agents on a system with multiple nodes, you would specify which components run on each agent resource. In this case, you have only one agent resource, so all the components run on this agent.

   a. On the same row as the `Prod JPetStore agents` resource, click **Actions > Add Component**.
      The **Actions** button is not displayed until you hover the mouse over the resource.

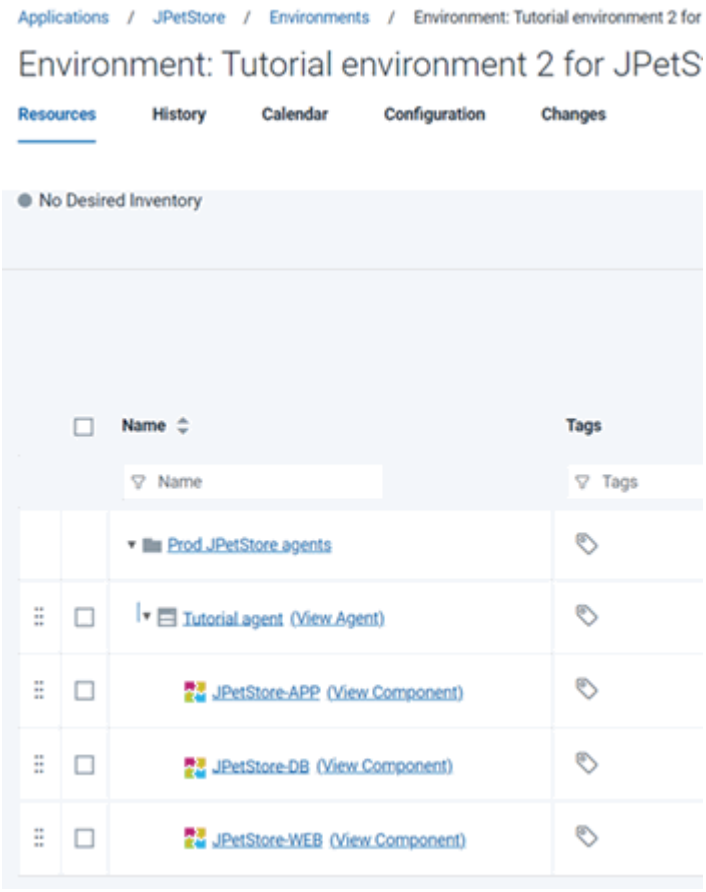
   b. Select the application component and then click **Save**.

   c. Repeat the process to add the other components to the resource.

   Note: Be sure to add each component to the agent resource and not to another component. The components must be at the same hierarchy level, and not child items of each other.

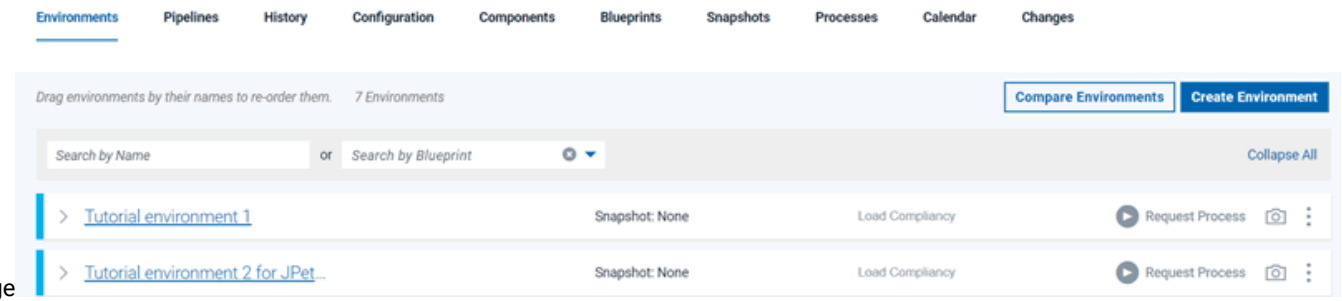
   Result

The components are mapped to the target system, as shown in the following figure:



Results

The **Environments** tab includes the new environment, as shown in the following image



With the environment prepared and the agent running, you can run the application process on the environment to deploy the components that you specified in the snapshot.

What to do next

In this lesson, you added an environment to the application

In the previous tutorial, you proved that the components in the first environment are compatible and complete. You can use the snapshot that you created to quickly promote all the components to the new environment. Your delivery pipeline might include deploying an application to testing and quality assurance servers before you deploy it to a production server. By using a snapshot, you can ensure that you quickly and correctly select all the right component versions as you move an application through the delivery pipeline.

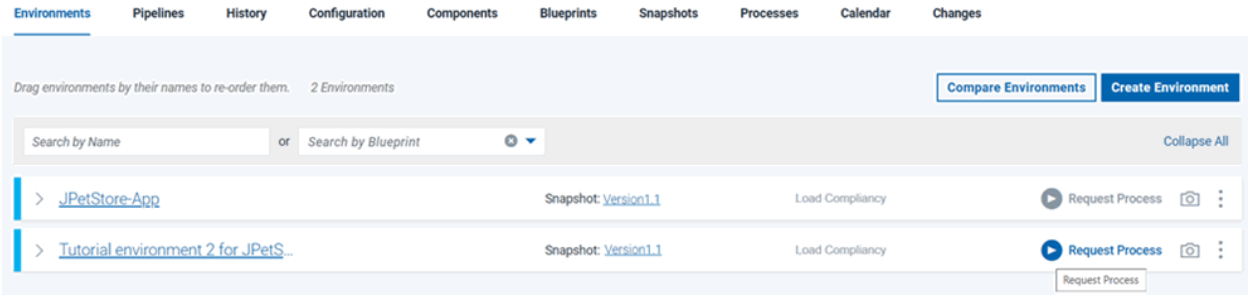
Lesson 3: Deploying the snapshots

To promote the snapshot to an environment, apply the snapshot when you run the application process.

About this task

In this lesson, you apply a snapshot to your new environment to quickly promote the components to a new environment. When you request the application process, you select the snapshot to deploy a selected set of components to an environment.

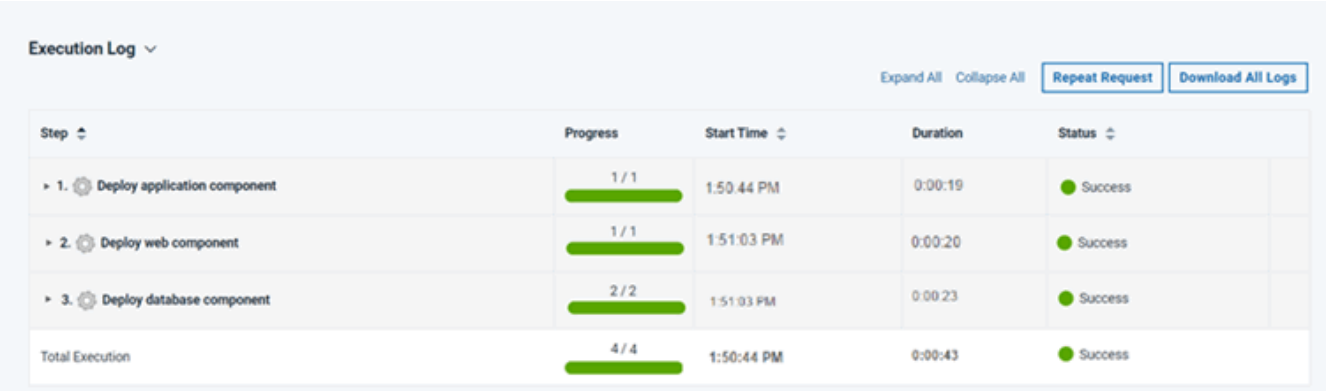
1. Open the application page by clicking **Applications** and then clicking `JPetStore`.
2. In the same row as your new environment, click the **Request Process** icon, as shown in the following figure.



3. In the **Run Process** window, in the **Process** list, select the `Deploy JPetStore` process.
4. In the **Snapshot** list, select the `Version 1.1` snapshot.
5. Click **Submit**.

Results

The web page shows you the progress of the application process request, and you can watch as the processes run. If the process finishes, the request shows that each component process is finished, as in the following figure:



You can view the running application at the following URL by substituting the host name of your target system for *hostname*. This is the host name of the agent that you configured in Configuring the agent and target system on page 1935. In the URL, specify the *tomcat.contextroot* value that you entered for this environment after the port number.

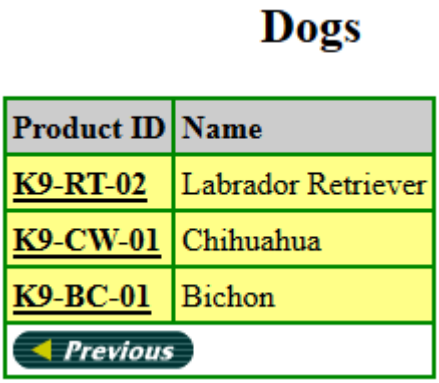
```
http://hostname:8081/JPetStoreProd
```

The application shows a simple online shopping site in the form of a pet store. The following figure shows the home page:

Welcome to JPetStore 4

Enter the Store

If you browse to the Dogs category, you see that the Bichon section is present. As you saw in the previous tutorial, this entry indicates that the JPetStore-DB component deployed both the 1.0 and 1.1 versions.



What to do next

This application process installed all components in the application by running their component processes. You defined the versions of the component processes in the snapshot.

You used a snapshot to quickly apply a compatible set of component processes to an application environment and viewed the deployed web page.

Summary

In this tutorial, you created and deployed a snapshot to an environment in HCL DevOps Deploy (Deploy).

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How to create a snapshot of the components of an application
- How to configure environments for deployments
- How to manage an application pipeline by moving specific application configurations from one environment to another

Advanced tutorials

Advanced tutorials for experienced users.

You can find tutorials in this section that cover particular use cases, such as the Blue-Green Deployment tutorial on page 1984.

Some tutorials in this section provide strategies for enterprise-scale scenarios with extensive resource requirements. Consequently, some tutorials abstractly simulate large-scale topologies. This approach means that you should be able to work through the tutorials by using a single system. Additionally, some tutorials provide a head-start by including resources that you can download or that build on other tutorials. The intermediate tutorial JPetStore tutorial on page 1934, for example, is referenced by several tutorials in this section.

Blue-green deployments

In this tutorial, you setup a deployment with two identical production environments, referred to arbitrarily as Blue and Green. One environment is always live, and the other environment is inactive and available to host an upgraded version.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Use tags to design resources for multiple-node environments Deploy.
- Configure environments for no-downtime deployments.
- Create application processes to deploy multiple components to multiple nodes.
- Configure user approvals for production deployments .

Time required

2-3 hours

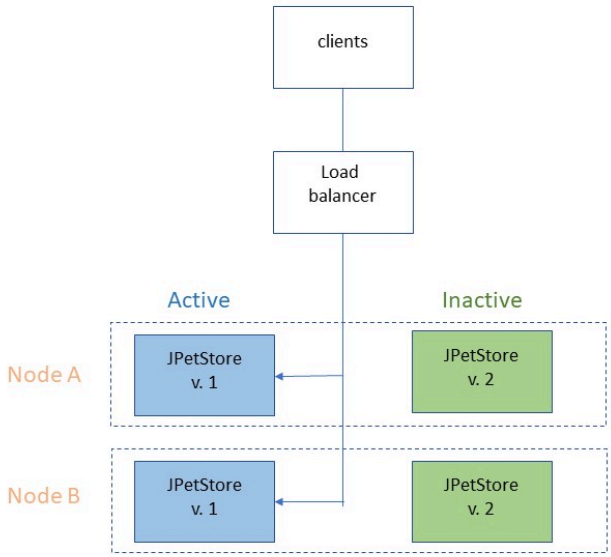
Introduction

In this tutorial, you setup a deployment with two identical production environments, referred to arbitrarily as Blue and Green. One environment is always live, and the other environment is inactive and available to host an upgraded version.

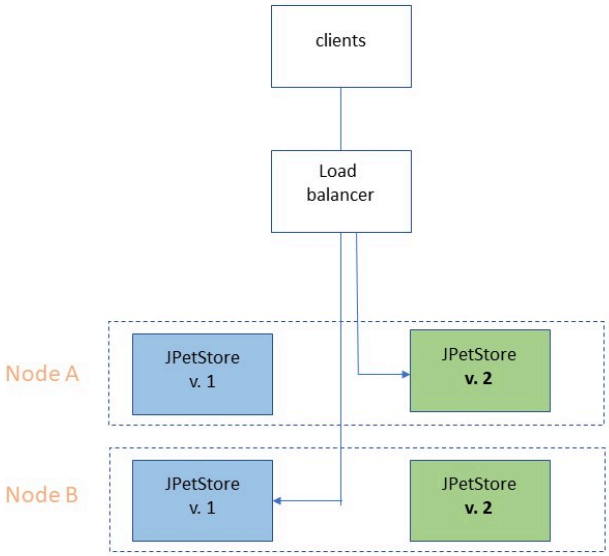
Business-critical multinode installations that must always remain in production require advanced updating strategies. *Blue-green deployments* fit this requirement because they provide smooth transitions between versions, zero-downtime deployments, and quick rollbacks.

A blue-green deployment uses two identical production environments. After verifying that an update in the inactive environment is working with a new version, you switch traffic to the new version. If anything goes wrong, you can quickly switch back to the old version.

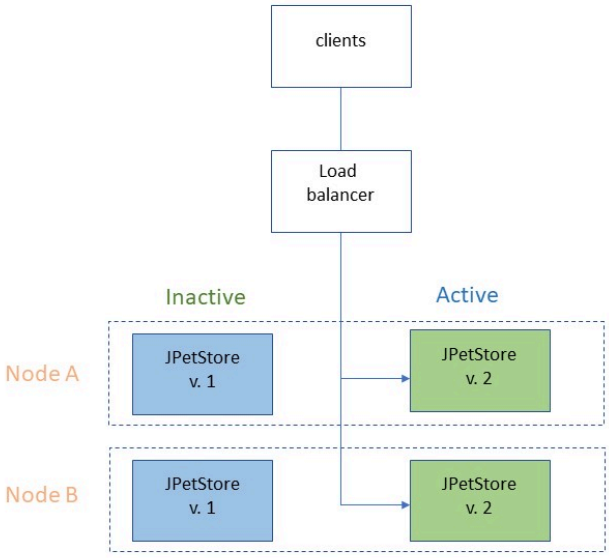
This graphic illustrates a two-node production topology. While customers use the nodes in the active blue environment, you update the inactive green environment with the new version. You can perform as many tests as required in the inactive green environment. When everything is ready, you only need to update the load balancer to switch production environments.



One approach is to switch traffic to a single updated node and validate it before switching the other nodes. Testing one node in this way is called a canary deployment. If the canary node fails, traffic remains uninterrupted because the non-updated nodes continue to serve customers.



If the update is successful, you direct the load balancer to switch all the remaining traffic to the other updated nodes.



When you are ready for the next release, you start the cycle over using the now-interactive blue environment.

Learning objectives

In this tutorial, you learn how to complete these tasks:

- Use tags to design resources for multiple-node environments Deploy.
- Configure environments for no-downtime deployments.
- Create application processes to deploy multiple components to multiple nodes.
- Configure user approvals for production deployments .

Time required

This tutorial takes 2 - 3 hours to finish. If you explore other concepts that are related to this tutorial, it could take longer to complete.

System requirements

To run this tutorial, you need the following resources:

- An installation of Deploy, version 8.2.0 or later.
- A Deploy agent installed in your environment.
- While not required, you can use multiple computers, virtual images, or cloud instances to use for agents in the tutorial.

Prerequisites

- This tutorial assumes familiarity with Deploy. If you new to the product, you might benefit from completing the Basic on page 1914 or Intermediate on page 1933 tutorials.
- You must have a Deploy account with the permissions to create and edit applications, environments, and components, and to create and run processes.
- This tutorial uses the application created in the JPetStore tutorial on page 1934. You can also download some of the resources created in the JPetStore tutorial. If you did not complete the JPetStore tutorial, you can use any multi-component application that you have sufficient permissions to update and modify.

Configuring the tutorial system

Configure the resources to define the blue and green systems.

Before you begin

Before you begin the tutorial, make sure that you complete the Prerequisites on page 1987.

About this task

Like all Deploy tutorials, to work through this one you need an agent that is running on a target computer system or systems. The target can be a cloud resource, a virtual image, or a physical computer.

1. The tutorial references the application created in the JPetStore tutorial on page 1992.
   If you did not complete that tutorial, you can use any multi-component Deploy application that you have sufficient permissions to modify.

2. While the examples in the tutorial show information for a Linux™ operating system, you can use any operating system.
3. Ensure that you have an agent installed in your environment.
   The way these logical environments are implemented relies on the way that Deploy represents target environments as resources. In the tutorial, you create a single production environment consisting of several nodes, and then duplicate the resources to represent the second system. These resources behave like separate systems, even though the resources point to the same physical system. For simplicity, you can use a single agent for the logical environments that you create in the tutorial.
4. Add component tags to the JPetStore components.

   You can tag components while you work on the **Components** tab, or you can tag components while you work in the **Resources** tab. Make sure that you use the **Components** tab because the tags that you associate with components in these two contexts behave differently.

   To add tags to components, complete the following steps:

   a. On the **Components** page, click the **Add tag** action for the JPet-Store-APP component, and then select the `JPetStore` tag.
      If you do not have the tag, in the **Add Tag** window, click **Create Tag**, and then define the new tag.

   b. Add the `JPetStore` tag to the JPet-Store-WEB, and JPet-Store-DB components.
      Later in the tutorial, you use the tag in an Install Multiple Components application process step on page 1992. For the step to work, the tagged component processes must all have the same name.

      Note: If you are using another multi-node application instead of JPetStore, add the same tag to each component used by the application.

   Result
   By using component tags, your resources and application can access all the components in an application with a single tag reference.

Creating the load balancer component

Create a component to abstractly represent the load balancer.

About this task
Multinode installations routinely use load balancers to rout incoming traffic to the next eligible node. There are many load balancers and just as many ways to configure them. For the tutorial, the load balancer is represented as a component within the JPetStore application.

The tutorial load balancer component has an operational process called `Change node port` that abstractly changes the active port on a node. You pass the port name that should be active during processing. You use the environment property `environment.LBPort` to define the logical environment's port.

1. Create a component and name it `Load balancer` or something similar.
2. Create an **Operational (No Version Needed)** process for the `Load balancer` component and name it `Change node port`, or something similar.
   You do not need versioning for the simple process that you create for the component.
3. Add the `Load balancer` component to the JPetSore application.

Results

The

Lesson 1 Define resources

Create agent resources for a multi-node deployment.

Before you begin

Before you begin the lesson, complete the Configuring the tutorial system on page 1987 topic.

About this task

In this tutorial you create two systems that are logically separate, but, for ease of use, run on the same hardware. The two environments behave in the same way and have the same dependencies.

To simulate separate physical environments, you can run the Blue and Green environments on the same server. If you have multiple servers available, of course you can use those. The Green resources run on one port, and the Blue resources run on another. This way, you can simply change the port to direct traffic to a different environment.

Deploy provides several loop-type steps that you can use to deploy multiple components. The For Each Agent loop, for instance, lets you control which agents run automated tasks. This way, even if you have a large number of agents, you can run tasks on specific agent groups by tagging agents and running tasks only on agents with those tags. The For Each Tag loop, which is similar, runs once for each of the resource tags that you specify. So you can organize the resources in your environment by tag, and then control which tags to use. For example, you can deploy to canary nodes first and all remaining nodes second, all in the same process.

Follow these steps to create the resource groups that represent the blue-green systems:

1. Create a top-level resource for the blue system.
   a. On the **Resources** page, click the **Resource Tree** tab, and then click **Create Top-Level Group**, and name the group `Blue System`.
   b. Add an agent resource to `Blue System` and name it `blue-A`
   c. Add the **JPetStore** component tag to `blue-A` agent resource.
      The single component tag references all three components in the JPetStore application.

Note: If you use you own application, make sure to tag the components and then use that tag.

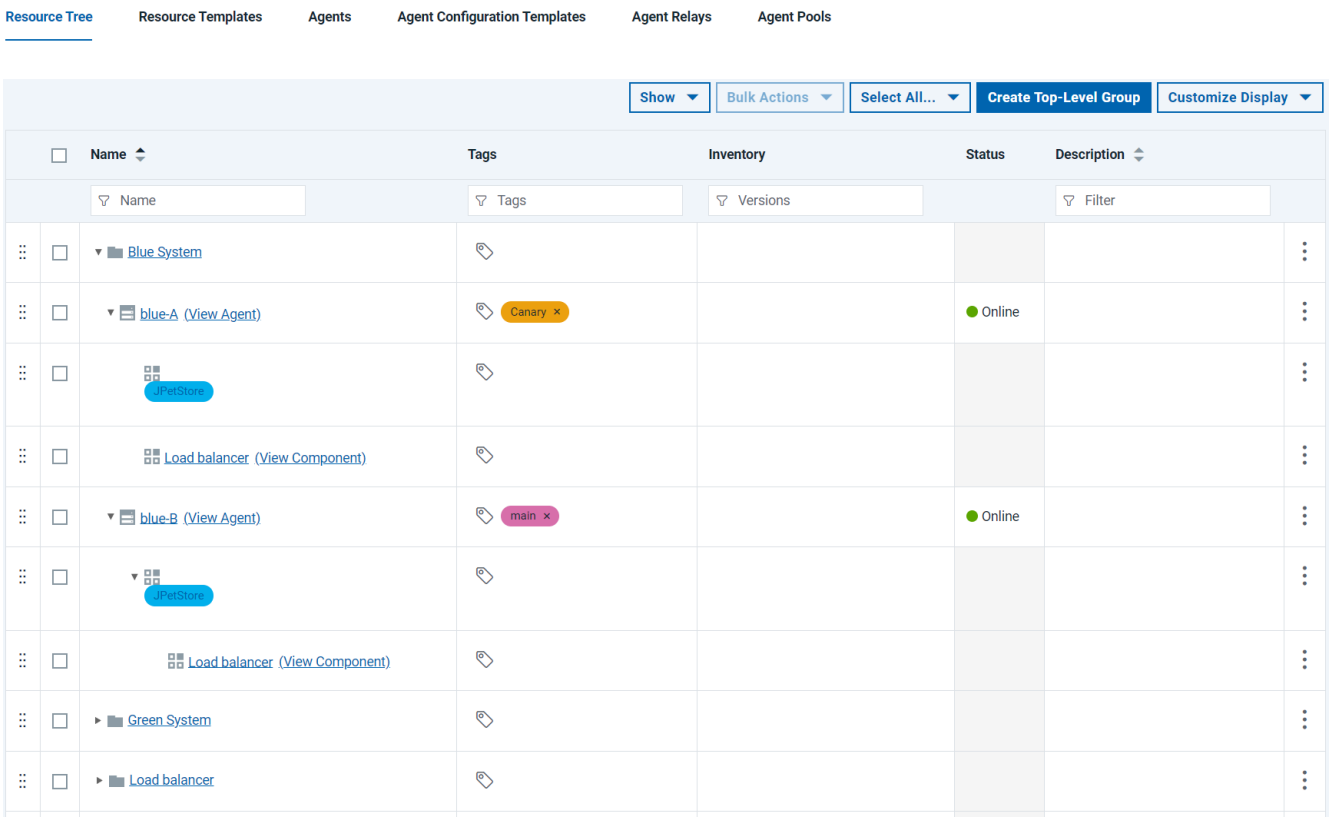
d. Create agent the `blue-B` agent resource and add to it the **JPetStore** component tag.

e. Add the `Load balancer` component to the `blue-A` and `blue-B` resources.

2. Click **Add tag** for the `blue-A` agent resource, and select the `Canary` tag, creating a new tag if necessary.
   The canary node always gets updated first. You update the other node when you're satisfied that the canary node is working properly.
3. Click **Add tag** for the `blue-B` agent resource, and select the `Main` tag, creating a new tag if necessary.
   All agent resources that are not Canary nodes use the `Main` tag.
4. Create a top-level resource for the green system by duplicating the actions in the previous steps.
5. Create a top-level resource for the load balancer, and add the `Load balancer` component to it.

Results

The complete resource trees look like the following figure:



In this lesson, you created agent resources required by the blue-green topology.

Lesson 2: Preparing the application

Prepare the application environments for the blue-green deployment.

Before you begin

Before you begin the lesson, make sure that you completed the Prerequisites on page 1987.

About this task

In this lesson, you modify the JPetStore application by adding the resources that you created in the previous lesson. To complete the lesson, you define environments and create properties that are used by the process that you create in the next lesson.

If you use another application other than JPetStore, you can still apply the steps described in this lesson.

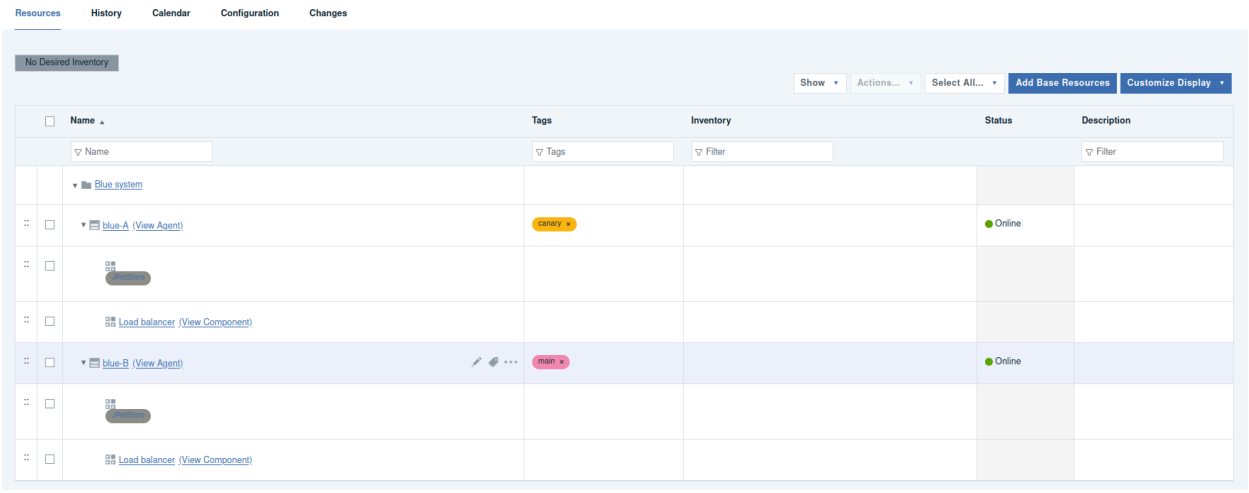
Creating the environments

Prepare the blue-green environments.

1. For the JPetStore application, create an application property and name it `activeEnvironment`, and assign it a value of `Blue environment`.
   The application process checks the value of this property to ensure that the active environment is not updated. At the start of the tutorial, the Blue environment is the active environment.
2. Create an environment, named Blue environment, to use for the Blue system.
3. Create an environment property for the Blue environment and name it `LBPort`. Assign a value of `8081` to the property.
   When you create component processes, you often define parameters that specify the location of artifacts on the target system, such as the location of the application server. Because these properties can vary on different target systems, you specify these properties as environment properties. Because you're using ports to distinguish the Blue and Green environments, you add a property to each environment named `LBPort`. This property stores the port that the application is running on in each environment; in this case, 8081 for Blue and 8082 for Green.

   Note: For the tutorial, you can use any environment property. The tutorial abstractly references ports because, as an exercise, you might use a proxy server to simulate a load balancer. Switching ports with a proxy server, such as HAProxy, is an easy way to test blue-green deployments without using an actual load balancer and separate nodes.

4. Add the JPetStore environment properties to the environment. on page 1958
   The JPetStore application uses several environment property to deploy its web application. If you are using your own application, you can skip this step.
5. Add the`Blue system` agent resource to the Blue environment.
   Because you put the `JPetStore` component tag on the resource, you don't have to map the individual components. The component tag grabs all of the components at once.
   Result
   The resource is shown on the **Resources** tab for the environment, as shown in the following figure:



6. Add the load balancer agent resource to the environment.
   Later, you call processes to update the load balancer as part of the regular deployment processes.
7. Create the Green environment similar to the Blue environment, and assign the Green system agent resource to it.
   Create the`LBPort` environment property for the Green environment, and assign the value `8082` to it.

Results

The new environments appear on the **Environments** tab.

Lesson 3: Create the application process

Create the application process used by the blue-green deployment.

Before you begin

Before you begin the tutorial, complete the Prerequisites on page 1987. Make sure to add the component tags to the JPetStore application's three components on page 1988.

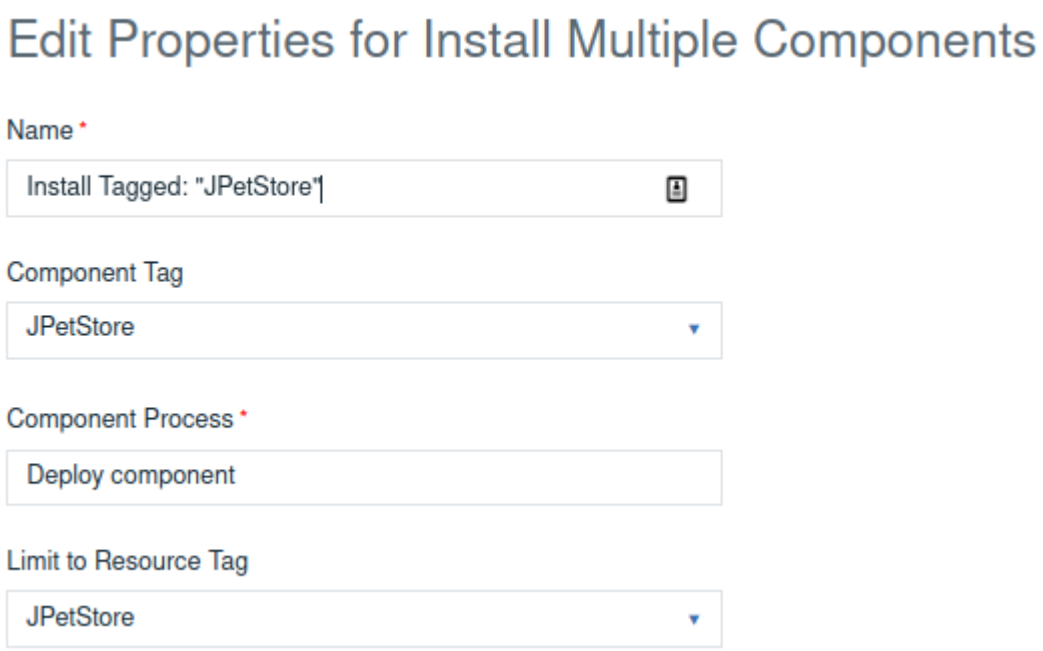
About this task

Now that you have the application, components, and environments set up, you design the automation for the Blue-Green deployment. The blue-green application deploys new version of the components, updates the load balancer, and verifies that everything is running properly.

1. Create a process for the JPetStore application, and name it `Blue-green`, or something similar.
2. On the process designer, add a **Switch** step that tests the `activeEnvironment` property.
   There are several ways to ensure that the process is running on the correct environment, but this method easy to implement. The Switch step compares on page 2400 the `activeEnvironment` property to the name of the environment on which the process is running. If those names match, that means the process is running on the active environment, which is not what you want and the process ends.

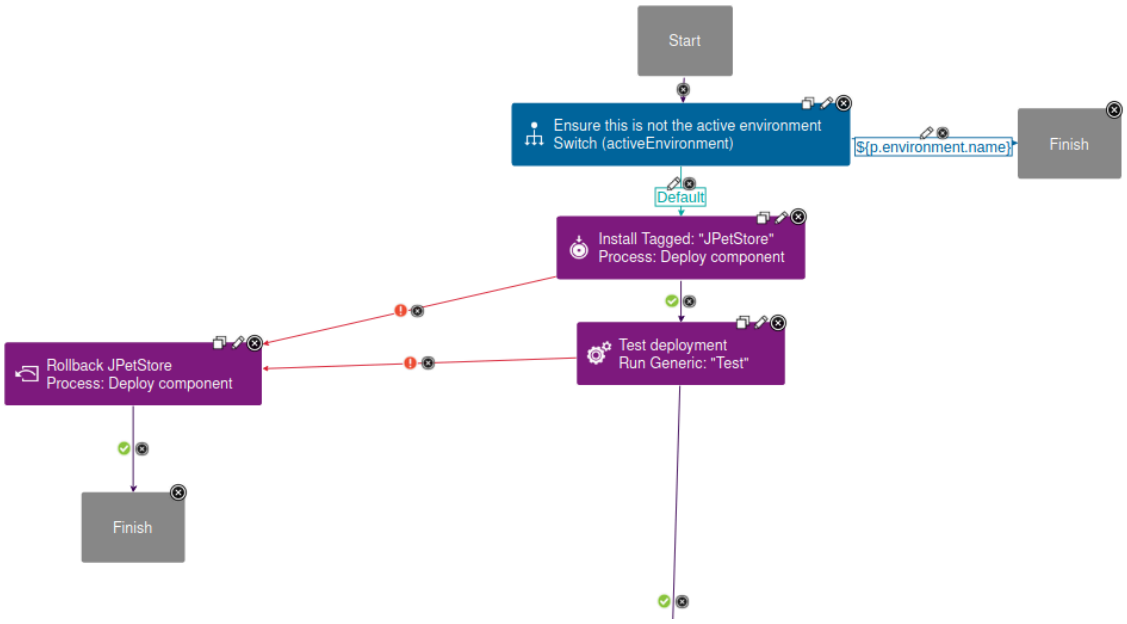
Note: If the `activeEnvironment` property is not on the **Property Name** list, review the previous lesson in Creating the environments on page 1991.

3. Add an **Install Multiple Components** step.
   This step uses the `JPetStore` tag to deploy the JPetStore components. This step iterates over each component with the specified tag. If you use your own application, ensure that the component processes all have the same name. The components are deployed to the `green-a` and `green-b` resources in the currently inactive Green system.



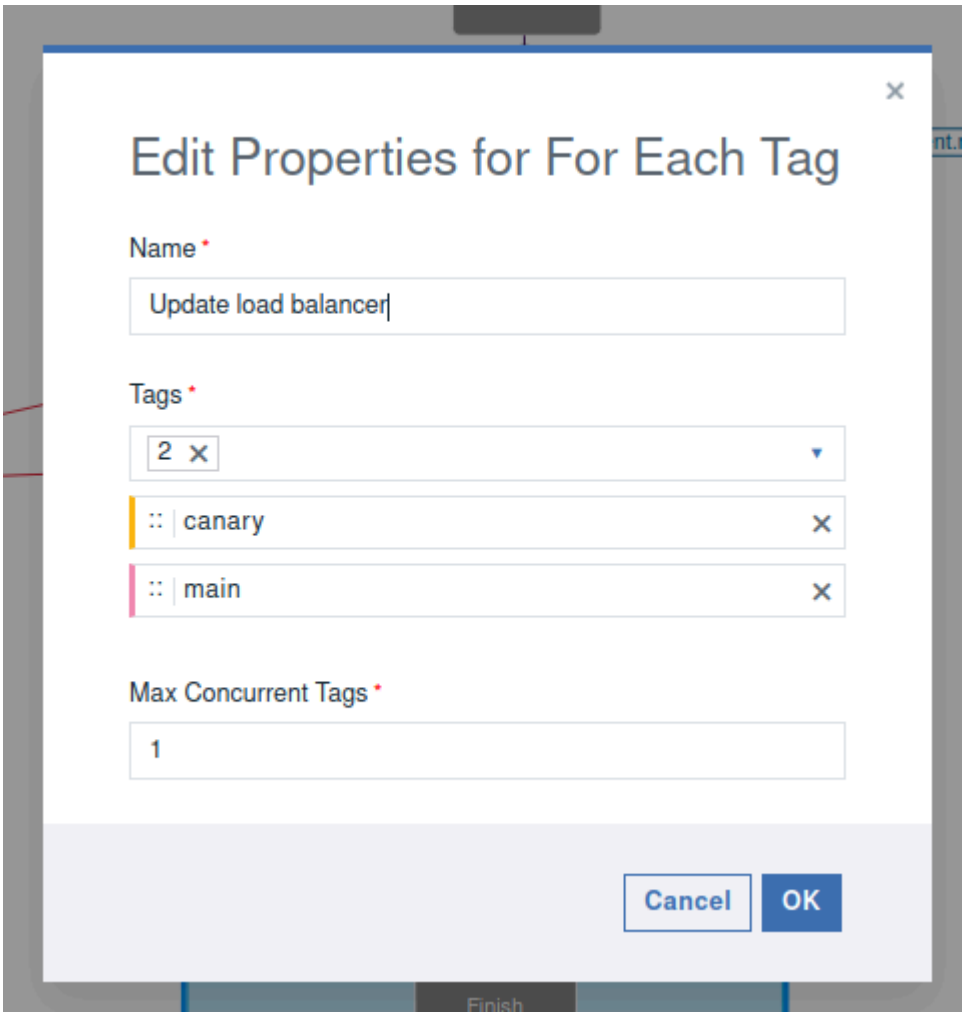
4. **Optional:** Add the optional automated testing step.
   If you typically run tests on your deployments, this is a good place to add them. For simplicity, you might use a Shell step or a generic process to as a placeholder for processing flow.
5. Add a **Rollback Multiple Components** step.
   If either the deployment step or the automated test fails, the process rolls back the newly deployed components to the version that was deployed at the beginning of this application process. As with the previous step, you might use a manual step or run a generic process step to represent the rollback feature.

   So far, the deployment is a typical multi-component deployment, and customers are unaware that an upgrade is in progress in the inactive system. In the next step, you define the **For Each Tag** loop that contains the primary blue-green processing.



6. Add a **For Each Tag** utility step to the For Each Tag loop.

   By the time processing gets to this loop, the new version of the application is already deployed to the inactive environment. The purpose of this loop is to update the load balancer in stages. In the loop, the `Canary` tag runs first, followed by the `Main` tag. This works because the For Each Tag step lets you restrict processing to specific tags and control the order in which they are processed. When the `Max Concurrent Tags` property is set to 1, only one node is processed at a time. In this case, the `Canary`-tagged node is processed before the other nodes.



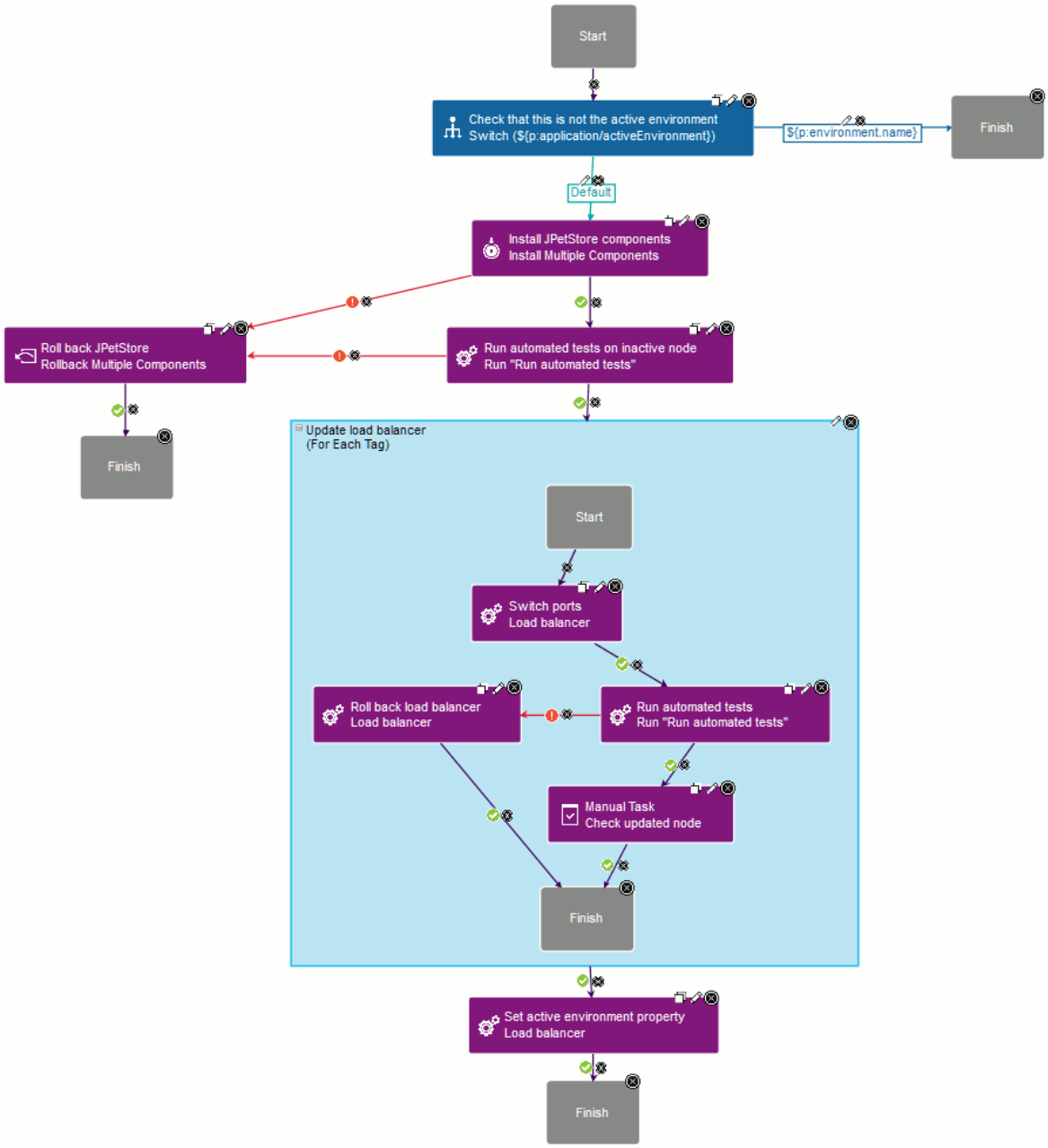
7. Within the loop, add the Load balancer component step.
   This step changes a property on the load balancer component that represents the active port on page 1988. The component process can access the properties it needs directly, without accepting any passed values.

   Now customers are using the new version of the software for the first time on the canary node. However, on the first time through the loop, the process updates the port for only the canary node.

8. In the loop, add a test step.
   The process runs automated tests on the node that is affected by the tag. If those tests fail, another step reverts the load balancer.
9. In the loop, add a rollback step.
   If the tests fail, this step reverts the load balancer to the last version.
10. Add a manual task step that you can use to verify the updated.

    The manual approval lets you verify that the canary node is working properly before the loop runs again and updates the other nodes. Your approval process is probably different, but the manual approval lets you verify that the deployment is running the way you expect for this example tutorial.

11. Click the new process to open it in the process editor.
    At the end of the process, the process updates the active environment property with the name of the newly active environment. I'm using a component process on the load balancer component for this, but there are other ways to change the application property.



Lesson 4: Deploying the application

Deploy the blue-green application.

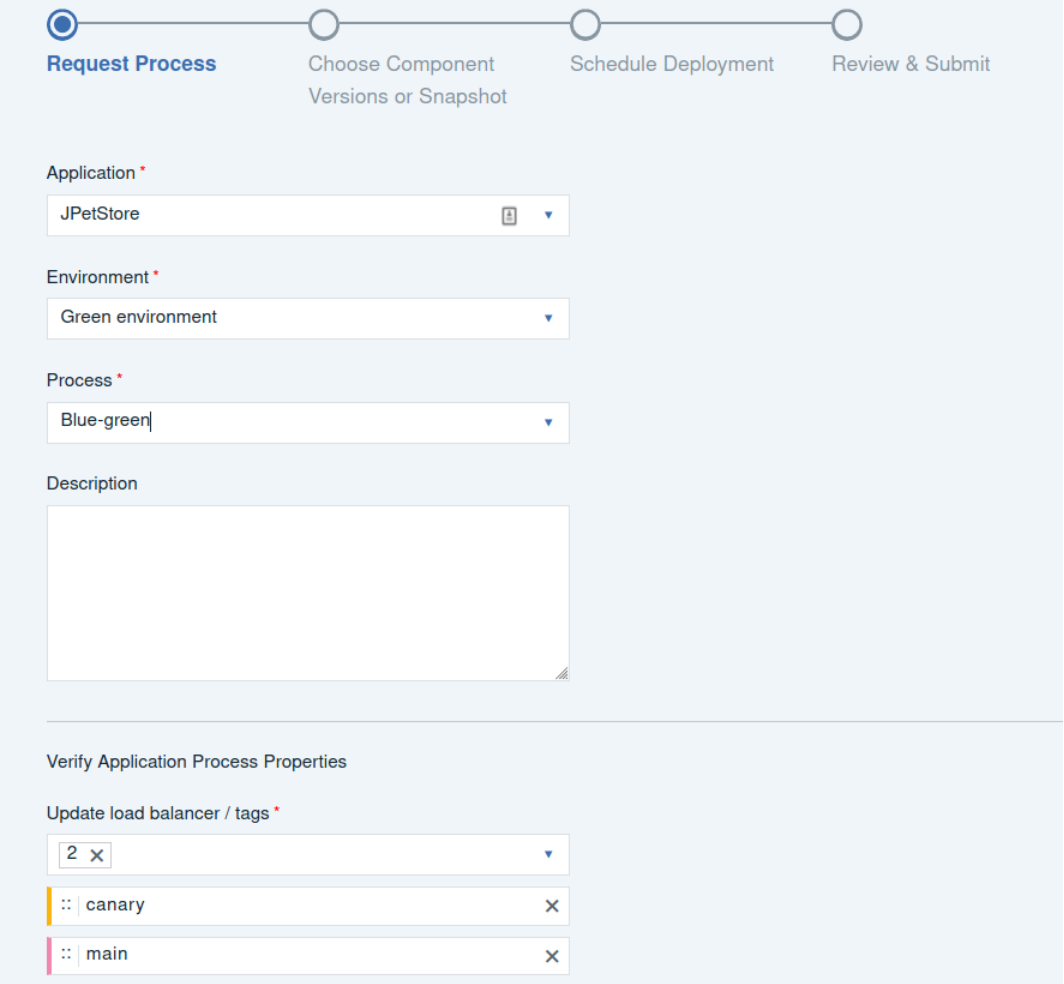
Before you begin

Before you begin the tutorial, complete the previous lesson on page 1992.

About this task

To deploy the blue-green application, run the application in the inactive environment. If you mistakenly run the application in the active environment, the process ends before any components are deployed. When the deployment finishes, the property that maintains the currently active environment is updated.

1. Run the deployment in the Green environment.
   The tutorial initially sets the active environment to the Blue environment.
2. On the **Request Process** page, select the process that you created for the tutorial, `Blue-green`.

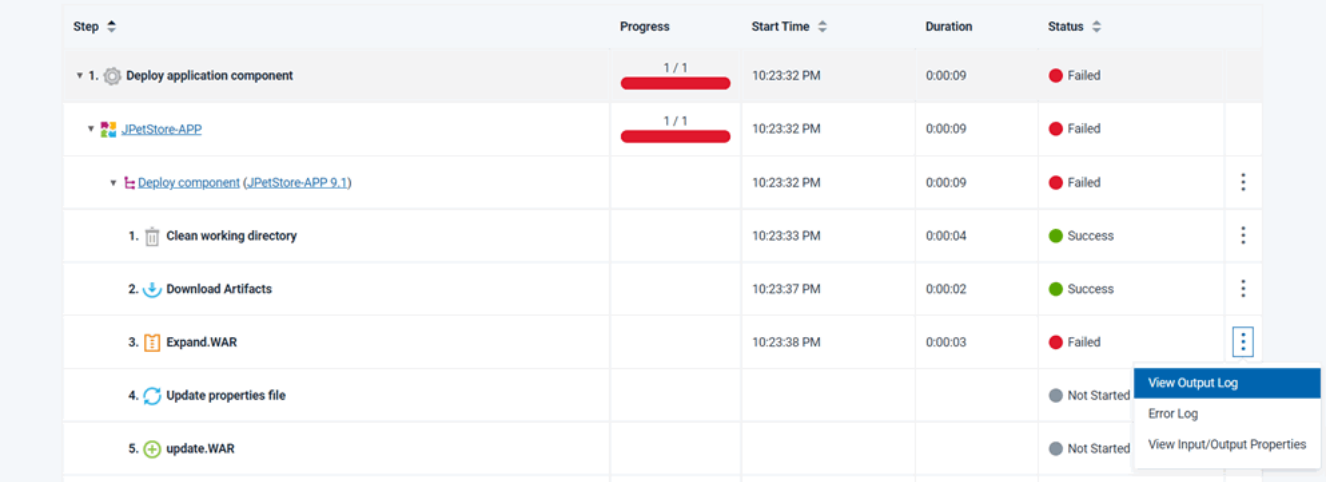


3. Go to **Choose Component Versions** and click **Select For All > Latest Available**.

Results

The web page shows you the progress of the application process request. From this page, you can watch as the processes run.

If the application process did not run successfully, one of the component processes might have failed. To troubleshoot this failed process, expand the log section for the component, and find the step that failed:



From here, you can look at the command-line output log for each step in the process by clicking the **Output Log** icon.

If one of your component processes did not complete, identify the step that did not complete. Verify that the properties for that step are correct and that the steps are in the correct order in the process. Then, run the application process again.

Summary

In this tutorial, you learned a strategy to manage blue-green deployments.

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How to model logical environments for *n* multi-node topologies
- How to incorporate a load balancer in a multi-node topology
- How to use component and resource tags in deployments
- How to use Switch steps
- How to use For All Tags loop to deploy multi-component applications

How-tos

Instructions for specific features.

This section provides instructions for specific product features, such as configuring webhooks.

How-to: Deploy applications by using REST commands

The following example describes how to use the REST commands to deploy an application.

Before you begin

This example follows the steps in the tutorial Create a simple helloWorld deployment on page 1915. Review the tutorial before you attempt to follow this example. Configure your system as described in Configure the helloWorld tutorial system on page 1916. Substitute values for your server name, user name, and password where appropriate.

About this task

This example shows the use of the Linux™ curl program to run REST commands. You can use many different programs or programming languages to run REST commands. The code authenticates with the server by specifying the user name and password on the command line. To learn more about running REST commands and authentication, see Running REST commands and Authenticating for REST commands.

The server that is used in this example is `server.example.com`. The user is `nsmith` and the password is `passwd`.

1. Create and import a component.

    a. Use the createComponent PUT command to create the component.

    Example

    ```
    curl -k -u nsmith:passwd
      https://server.example.com:8443/cli/component/create
      -X PUT -d @newComponent.json
    ```

    Note: You must enter commands on a single line. Commands are split onto different lines here for clarity.

    The following JSON file represents the component:

    ```
    {
      "name": "helloWorld",
      "description": "",
      "importAutomatically": false,
      "useVfs": true,
      "sourceConfigPlugin": "File System (Versioned)",
      "defaultVersionType": "FULL",
      "properties": {

      "FileSystemVersionedComponentProperties/basePath": "/home/user1/artifacts/shared/hell
    oWorld"
      }
    }
    ```

    Result

    The command returns a result that is similar to the following JSON information:

    ```
    {
      "id":"cf355787-a772-4533-be45-3e169e9da6eb",
      "securityResourceId":"23504603-ab8c-4740-bc2d-d18c08f790e9",
      "name":"helloWorld",
      "description":"",
      "created":1396878409753,
      "importAutomatically":false,
      "useVfs":true,
    ```

```
  "active":true,
  "integrationFailed":false,
  "deleted":false,
  "defaultVersionType":"FULL",
  "cleanupDaysToKeep":0,
  "cleanupCountToKeep":0,
  "tags":[],
  "user":"nsmith"
}
```

b. Use the setComponentProperty PUT command to set a component property.

Example

```
curl -k -u nsmith:passwd
  "https://server.example.com:8443/cli/
  component/propValue?component=helloWorld&name=helloHome&value=c%3A%5ChelloWorld" -X PUT
```

Result

The command returns a result similar to the following JSON information:

```
{
  "id":"194b6e0c-3f30-4d9a-b3cf-dd61dd1a7328",
  "name":"helloHome",
  "value":"c:\\helloWorld",
  "secure":false
}
```

c. Use the importVersions PUT command to import the component version.

Example

```
curl -k -u nsmith:passwd
  https://server.example.com:8443/cli/component/integrate
  -X PUT -d {"component":"helloWorld"}
```

Result

The command returns the following result:

```
{}
```

2. Use the createComponentProcess PUT command to create a component process.

Example

```
curl -k -u nsmith:passwd
  https://server.example.com:8443/cli/componentProcess/create
  -X PUT -d @componentProcess.json
```

The following JSON file represents this component process:

```
{
  "active": "true",
  "component": "helloWorld",
  "configActionType": "ADD",
  "defaultWorkingDir": "${p:resource/work.dir}/${p:component.name}",
  "description": "",
  "inventoryActionType": "ADD",
  "name": "helloProcess",
```

```
  "propDefs": [],
  "rootActivity": {
    "children": [
      {
        "allowFailure": false,
        "children": [],
        "commandName": "Delete Files and Directories",
        "impersonationUseSudo": false,
        "name": "Clean work space",
        "pluginName": "File Utils",
        "pluginVersion": 31,
        "properties":{
            "baseDir":".",
            "includes":"*",
            "excludes":"",
            "followSymlinks":"false",
            "caseSensitive":"true"
         },
        "showHidden": false,
        "type": "plugin",
        "useImpersonation": false
      },
      {
        "allowFailure": false,
        "children": [],
        "commandName": "Download Artifacts",
        "impersonationUseSudo": false,
        "name": "Download Artifacts",
        "pluginName": "IBM UrbanCode Deploy Versioned File Storage",
        "pluginVersion": 18,
        "properties":{
           "directoryOffset":".",
           "artifactSetBaseDir":"",
           "fileIncludePatterns":"**/*",
           "fileExcludePatterns":"",
           "syncMode":"true",
           "fullVerification":"true",
           "setFileExecuteBits":"false",
           "verifyFileIntegrity":"false",
           "repositoryUrl":"${p:server.url}/vfs",
           "repositoryId":"${p:component/code_station/repository}",
           "label":"${p:version.name}",
           "serverUrl":"${p:server.url}",
           "compId":"${p:component.id}",
           "resId":"${p:resource.id}",
           "charset":""
        },
        "showHidden": false,
        "type": "plugin",
        "useImpersonation": false
      },
      {
```

```
        "allowFailure":false,
        "children": [],
        "commandName":"Create .zip File",
        "impersonationUseSudo":false,
        "name":"Compress artifacts",
        "pluginName":"File Utils",
        "pluginVersion":31,
        "properties":{
           "zipName":"helloWorld.zip",
           "baseDir":".",
           "includes":"*",
           "excludes":"",
           "update":"false",
           "followSymlinks":"false",
           "caseSensitive":"true"
        },
       "showHidden": false,
       "type": "plugin",
       "useImpersonation": false
     },
     {
        "allowFailure":false,
        "children": [],
        "commandName":"Move Directory",
        "impersonationUseSudo":false,
        "name":"Deploy artifacts",
        "pluginName":"File Utils",
        "pluginVersion":31,
        "properties":{
           "sourceDir":".",
           "destDir":"${p:helloHome}/target",
           "includes":"*.zip",
           "excludes":"",
           "mapperRules":""
        },
       "showHidden": false,
       "type": "plugin",
       "useImpersonation": false
     },
     {
       "children": [],
       "name": "FINISH",
       "type": "finish"
     }
   ],
   "edges": [
     {
       "to": "Clean work space",
       "type": "ALWAYS",
       "value": ""
     },
     {
```

```
      "from": "Clean work space",
      "to": "Download Artifacts",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "from": "Download Artifacts",
      "to": "Compress artifacts",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "from": "Compress artifacts",
      "to": "Deploy artifacts",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "from": "Deploy artifacts",
      "to": "FINISH",
      "type": "ALWAYS",
      "value": ""
    }
  ],
  "name": "GRAPH",
  "offsets": [
    {
      "name": "Clean work space",
      "x": "-65",
      "y": "90"
    },
    {
      "name": "Download Artifacts",
      "x": "-70",
      "y": "210"
    },
    {
      "name": "Compress artifacts",
      "x": "-70",
      "y": "290"
    },
    {
      "name": "Deploy artifacts",
      "x": "-65",
      "y": "370"
    },
    {
      "name": "FINISH",
      "x": "-60",
      "y": "450"
    }
  ],
```

```
    "type": "graph"
  },
  "status": "Active",
  "takesVersion": "true"
}
```

Result

The command returns a result that is similar to the following JSON information:

```
{
  "id":"12f65517-5fc1-44ff-a58b-2401cfba8f2f",
  "name":"helloProcess",
  "description":"",
  "defaultWorkingDir":"${p:resource\/work.dir}\/${p:component.name}",
  "takesVersion":true,
  "inventoryActionType":"ADD",
  "status":"Active",
  "active":true,
  "versionCount":1,
  "version":1,
  "commit":0,
  "path":"components\/cf355787-a772-4533-be45-3e169e9da6eb\/processes\/12f65517-5fc1-44ff-a58
b-2401cfba8f2f"
}
```

3. Create an application, and then associate the component with the application.

    a. Use the createApplication PUT command to create an application.

    Example

    ```
    curl -k -u nsmith:passwd
      https://server.example.com:8443/cli/application/create
      -X PUT -d @newApplication.json
    ```

    The following JSON file represents the application:

    ```
    {
      "name": "hello Application",
      "description": "",
      "notificationScheme": "None",
      "enforceCompleteSnapshots": "false"
    }
    ```

    Result

    The command returns a result that is similar to the following JSON information:

    ```
    {
      "id":"932452cc-3562-4545-8de5-c57aa7efbcc4",
      "securityResourceId":"2bc72218-dbfe-4c11-b16f-752b7ca63d06",
      "name":"hello Application",
      "description":"",
      "created":1396882026943,
      "enforceCompleteSnapshots":false,
      "active":true,
      "tags":[],
    ```

```
  "user":"nsmith"
}
```

b. Use the addComponentToApp PUT command to associate the component with the application.

Example

```
curl -k -u nsmith:passwd
  "https://server.example.com:8443/cli/
  application/addComponentToApp?component=helloWorld&application=hello%20Application"
  -X PUT
```

Result

The command returns the following result:

```
{}
```

4. Create an environment, and associate it with a resource.

a. Use the createEnvironment PUT command to create an environment.

Example

```
curl -k -u nsmith:passwd

 "https://server.example.com:8443/cli/environment/createEnvironment?application=hello%20Applicati
on&name=helloDeploy"
  -X PUT
```

Result

The command returns the ID of the environment, which is similar to the following string:

```
2142c0e3-7ebe-4a2f-9e0d-14ccacc70a79
```

b. Use the addEnvironmentBaseResources PUT command to associate the resource that you configured earlier with the environment.

Example

```
curl -k -u nsmith:passwd

 "https://server.example.com:8443/cli/environment/addBaseResource?application=hello%20Application
&environment=helloDeploy&resource=%2FhelloWorld%20Tutorial"
  -X PUT
```

Result

This command does not return a result if the resource is correctly associated with the environment.

c. Use the createResource PUT command to map the helloWorld component to the agent resource.

Example

```
curl -k -u nsmith:passwd
  https://server.example.com:8443/cli/resource/create
  -X PUT -d @componentResource.json
```

The following JSON file represents the component resource:

```
{
  "roleId":"09456abd-65e3-469e-94c6-a6a873b971a8",
  "name":"helloWorld",
```

```
  "description":"",
  "inheritTeam":"true",
  "useImpersonation":"false",
  "roleProperties":{},
  "parentId":"b2ff919e-b4c0-4d13-8973-eff44e4835e2",
  "teamMappings":[]
}
```

Result

The command returns a result that is similar to the following JSON information:

```
{
  "id":"ba958487-fb86-4290-9240-da766af37963",
  "securityResourceId":"66685b67-21ab-411d-99bb-f577a74d88b3",
  "name":"helloWorld",
  "path":"\/helloWorld Tutorial\/fit-vm11-157\/helloWorld",
  "active":true,
  "description":"",
  "inheritTeam":true,
  "impersonationPassword":"****",
  "impersonationUseSudo":false,
  "impersonationForce":false,
  "type":"subresource",
  "status":"ONLINE",
  "hasAgent":true,
  "tags":[]
}
```

d. Use the addTagToResource PUT command to add a tag to the component resource.

Example

```
curl -k -u nsmith:passwd
  "https://server.example.com:8443/cli/
  resource/tag?resource=ba958487-fb86-4290-9240-da766af37963&tag=blueCycle&color=0000FF"
  -X PUT
```

Result

This command does not return a result if the component is correctly tagged.

5. Use the createApplicationProcess PUT command to create an application process.

Example

```
curl -k -u nsmith:passwd
  https://server.example.com:8443/cli/applicationProcess/create
  -X PUT -d @applicationProcess.json
```

The following JSON file represents the application process. The application ID is the same as the ID that is returned in step 3.

```
{
  "application": "932452cc-3562-4545-8de5-c57aa7efbcc4",
  "description": "",
  "inventoryManagementType": "AUTOMATIC",
  "name": "hello App Process",
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
```

```
"rootActivity": {
  "children": [
    {
      "children": [{
        "children": [{
          "allowFailure": false,
          "children": [],
          "componentName": "helloWorld",
          "componentProcessName": "helloProcess",
          "name": "Deploy helloWorld",
          "properties": {},
          "type": "componentProcess"
        }],
        "componentName": "helloWorld",
        "name": "inventoryVersionCheck",
        "status": "Active",
        "type": "inventoryVersionDiff"
      }],
      "componentName": "helloWorld",
      "failFast": "false",
      "maxIteration": "-1",
      "name": "Deploy helloWorld",
      "runOnlyOnFirst": "false",
      "type": "componentEnvironmentIterator"
    },
    {
      "children": [],
      "name": "FINISH",
      "type": "finish"
    }
  ],
  "edges": [
    {
      "to": "Deploy helloWorld",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "from": "Deploy helloWorld",
      "to": "FINISH",
      "type": "ALWAYS",
      "value": ""
    }
  ],
  "name": "GRAPH",
  "offsets": [
    {
      "name": "Deploy helloWorld",
      "x": "-35",
      "y": "210"
    },
    {
```

```
        "name": "FINISH",
        "x": "0",
        "y": "420"
      }
    ],
    "propDefs": [],
    "type": "graph"
  }
}
```

Result

The command returns a result that is similar to the following JSON information:

```
{
  "id":"37d563ce-7885-48f1-b8c0-f79046c0f024",
  "name":"hello App Process",
  "description":"",
  "active":true,
  "inventoryManagementType":"AUTOMATIC",
  "offlineAgentHandling":"PRE_EXECUTION_CHECK",
  "versionCount":1,
  "version":1,
  "commit":0,
  "path":"applications\/932452cc-3562-4545-8de5-c57aa7efbcc4\/processes\/37d563ce-7885-48f1-b
8c0-f79046c0f024"
}
```

6. Use the requestApplicationProcess PUT command to deploy the application.

   Example

   ```
   curl -k -u nsmith:passwd
     https://server.example.com:8443/cli/applicationProcessRequest/request
     -X PUT -d @applicationDeploy.json
   ```

   The following JSON file represents the deployment request:

   ```
   {
     "application": "hello Application",
     "applicationProcess": "hello App Process",
     "environment": "helloDeploy",
     "versions": [{
       "version": "1.0",
       "component": "helloWorld"
     }]
   }
   ```

   Result

   The command returns a result that is similar to the following JSON information:

   ```
   {"requestId":"bffff82f-ff69-4c07-b010-26ca3f21be39"}
   ```

How-to: Export and import elements with YAML configuration

The following example task describes how to export and import elements with YAML configuration.

Before you begin

This example follows the steps in the Create a simple helloWorld deployment on page 1915 tutorial. Review the tutorial before you attempt to follow this example. Configure your system as described in Configure the helloWorld tutorial system on page 1916.

About this task

In this example, you export components in YAML format, configure, and import them to other HCL DevOps Deploy (Deploy)systems. However, you can export other elements such as applications, application templates, components, component templates, processes, and so on.

Exporting or importing of elements in YAML format is the same as in JSON format. The difference is that you need to specify appropriate headers in the REST calls. For APIs that take input, like import or save actions, you must add the following header: `Accept: application/yaml`. For APIs that produce output, like export and get actions, you must add the following header: `Content-Type: application/yaml`.

Note:

- If you don't use the headers, the elements are exported or imported in JSON format by default.
- Exporting and importing snapshots with YAML is not supported.

1. Export the helloWorld component you created in Lesson 1 Create the helloWorld component on page 1918 by using the REST command. See Export a component to a json file. for command reference.
   While entering the REST command, ensure that you use the following header with your request:

   ```
   Content-Type: application/yaml
   ```

   Result

   The command returns a result similar to the following YAML definition:

   ```yaml
   ---
   name: "helloWorld"
   description: ""
   created: 1700804083300
   componentType: "STANDARD"
   ignoreQualifiers: 0
   importAutomatically: false
   useVfs: true
   active: true
   deleted: false
   defaultVersionType: "FULL"
   cleanupDaysToKeep: 0
   cleanupCountToKeep: 0
   sourceConfigPluginName: "File System (Versioned)"
   envPropDefs: []
   resPropDefs: []
   versionPropDefs: []
   propSheetGroup:
       propSheets:
   ```

```
        - name: "template"
          properties: []
        - name: "custom"
          properties:
              - name: "helloHome"
                value: "C:\\Users\\Administrator\\Downloads\\helloWorld"
                description: ""
                secure: false
        - name: "FileSystemVersionedComponentProperties"
          properties:
              - name: "saveFileExecuteBits"
                value: "false"
                description: ""
                secure: false
              - name: "extensions"
                description: ""
                secure: false
              - name: "includes"
                description: ""
                secure: false
              - name: "excludes"
                description: ""
                secure: false
              - name: "basePath"
                value: "C:\\Users\\Administrator\\Downloads\\helloWorld"
                description: ""
                secure: false
processes:
    - name: "helloProcess"
      description: ""
      defaultWorkingDir: "${p:resource/work.dir}/${p:component.name}"
      takesVersion: true
      inventoryActionType: "ADD"
      status: "Active"
      configActionType: "ADD"
      active: true

  path: "components/18bffd2b-e64b-6d96-3016-4d13b84617e1/processes/18bffd9b-473b-6bc9-4e90-ad
03192b1cf4"
      deleted: false
      propDefs: []
      rootActivity:
          edges:
              - to: "Download Artifacts"
                from: "Clean work space"
                type: "SUCCESS"
                value: ""
              - to: "Clean work space"
                type: "ALWAYS"
                value: ""
              - to: "Deploy artifacts"
                from: "Compress artifacts"
```

```
            type: "SUCCESS"
            value: ""
        - to: "069538fb309ede5cd10eaf7c7c81e3"
            from: "Deploy artifacts"
            type: "SUCCESS"
            value: ""
        - to: "Compress artifacts"
            from: "Download Artifacts"
            type: "SUCCESS"
            value: ""
    offsets:
        - name: "Download Artifacts"
            x: -145
            "y": 310
            h: 110
            w: 440
        - name: "069538fb309ede5cd10eaf7c7c81e3"
            x: -5
            "y": 790
            h: 100
            w: 160
        - name: "Deploy artifacts"
            x: -120
            "y": 630
            h: 110
            w: 390
        - name: "Clean work space"
            x: -205
            "y": 150
            h: 110
            w: 560
        - name: "Compress artifacts"
            x: -120
            "y": 470
            h: 110
            w: 390
    layoutMode: "auto"
    type: "graph"
    name: "83dd413d-d8d6-4c45-8458-bc7f2e976a5e"
    children:
        - type: "finish"
            name: "069538fb309ede5cd10eaf7c7c81e3"
            children: []
        - allowFailure: false
            useImpersonation: false
            showHidden: false
            impersonationUseSudo: false
            pluginVersion: 96
            commandName: "Delete Files and Directories"
            pluginName: "File Utils"
            pluginId: "com.urbancode.air.plugin.FileUtils"
            authTokenRestriction: "20000001-0001-0001-0001-000000000000"
```

```
      properties:
          baseDir: "."
          includes: "*"
          excludes: ""
          followSymlinks: "false"
          caseSensitive: "true"
      cachedAuthTokenRestriction:
          name: "System Default"
          description:
              "Used by default for any plugin steps without a specified token\
              \ restriction"
          restrictions:
              - verb: "ALL"
                url: "/*"
      type: "plugin"
      name: "Clean work space"
      children: []
    - allowFailure: false
      useImpersonation: false
      showHidden: false
      impersonationUseSudo: false
      pluginVersion: 46
      commandName: "Download Artifacts"
      pluginName: "UrbanCode Deploy Versioned File Storage"
      pluginId: "com.urbancode.plugin.VFS"
      authTokenRestriction: "20000001-0001-0001-0001-000000000000"
      properties:
          directoryOffset: "."
          artifactSetBaseDir: ""
          fileIncludePatterns: "**/*"
          fileExcludePatterns: ""
          syncMode: "true"
          handleIncrementalVersions: "false"
          fullVerification: "true"
          setFileExecuteBits: "false"
          verifyFileIntegrity: "false"
          charset: ""
          versionId: "${p:version.id}"
          versionType: "${p:version.type}"
          serverUrl: "${p:server.url}"
          compId: "${p:component.id}"
          resId: "${p:resource.id}"
          envId: "${p:environment.id}"
          maxMemory: "1G"
          label: ""
      cachedAuthTokenRestriction:
          name: "System Default"
          description:
              "Used by default for any plugin steps without a specified token\
              \ restriction"
          restrictions:
              - verb: "ALL"
```

```
                    url: "/*"
              type: "plugin"
              name: "Download Artifacts"
              children: []
            - allowFailure: false
              useImpersonation: false
              showHidden: false
              impersonationUseSudo: false
              pluginVersion: 96
              commandName: "Create .zip File"
              pluginName: "File Utils"
              pluginId: "com.urbancode.air.plugin.FileUtils"
              authTokenRestriction: "20000001-0001-0001-0001-000000000000"
              properties:
                  zipName: "helloWorld.zip"
                  baseDir: "."
                  includes: "*"
                  excludes: ""
                  update: "false"
                  followSymlinks: "false"
                  caseSensitive: "true"
                  customEncoding: ""
              cachedAuthTokenRestriction:
                  name: "System Default"
                  description:
                      "Used by default for any plugin steps without a specified token\
                      \ restriction"
                  restrictions:
                      - verb: "ALL"
                        url: "/*"
              type: "plugin"
              name: "Compress artifacts"
              children: []
            - allowFailure: false
              useImpersonation: false
              showHidden: false
              impersonationUseSudo: false
              pluginVersion: 96
              commandName: "Move Directory"
              pluginName: "File Utils"
              pluginId: "com.urbancode.air.plugin.FileUtils"
              authTokenRestriction: "20000001-0001-0001-0001-000000000000"
              properties:
                  sourceDir: "."
                  destDir: "${p:helloHome}/target"
                  includes: "*.zip"
                  excludes: ""
                  mapperRules: ""
              cachedAuthTokenRestriction:
                  name: "System Default"
                  description:
                      "Used by default for any plugin steps without a specified token\
```

```
                        \ restriction"
                    restrictions:
                        - verb: "ALL"
                          url: "/*"
                type: "plugin"
                name: "Deploy artifacts"
                children: []
configTemplates: []
genericProcesses: []
teamMappings: []
tags: []
```

2. Copy the content into a text editor and save it.
3. Edit the YAML configuration to include the changes you require.
4. Import the component in YAML format by using REST command to the destination Deploy server.
   See Import component from json file. for command reference. You must use the following header with the REST call.

   ```
   Accept: application/yaml
   ```

Results

The file is imported and the record is displayed in the Components list.

How-to: Integrate MS Teams

Integrate Microsoft Teams with HCL DevOps Deploy (Deploy) to post notifications automatically to your Teams channel whenever a specific deployment event happens.

Before you begin

- A process such as a generic process is created. See Processes on page 2398.
- Create a Webhook URL from your Teams channel. Refer to the Microsoft Teams documentation.

About this task

By using the Incoming Webhook connector in Teams, you can integrate Deploy services so that you get notified of an event through the Teams channels.

In this example, you will learn about the requirements and procedures to integrate Teams with Deploy. To integrate Teams with Deploy, follow the steps provided in this section.

Step 1: Format the message card

A message card is defined in a JSON format that includes the message in text, links, and image, if required. For HCL DevOps Deploy (Deploy), you can create a message card that refers to the process name, status, and a link to the process deployment page. The message card format is used as body in the webhook template.

About this task

The message cards can be created using the following available templates:

- Application Deployment Failure JSON body on page 2016
- Application Deployment Success JSON body on page 2019
- Application Process Request Started JSON body on page 2022
- Approval Completed JSON body on page 2022
- Approval Failed JSON body on page 2023
- External Approval Completed JSON body on page 2024
- External Approval Failed JSON body on page 2025
- Generic Process Completed JSON body on page 2026
- Generic Process Request Started JSON body on page 2026
- Scheduled Process Not Started JSON body on page 2027

Step 2: Create a webhook template

Create a webhook template for an event so that when the event happens a notification is triggered on Teams.

1. Select **Settings > Webhook Templates > Create Webhook Template**.
2. Provide a name and description for the template. For this example, give the name as `Process failure notification`.
3. Click on **Body** area and paste the example code from the step 1.
4. Choose the event type **Process Failure** from the **Type** field.
5. Submit your changes.

Results

The webhook template is created and ready to be added to an application.

Step 3: Add webhook template to an application

You can add a webhook template to applications, generic processes, and external approval processes. For this example, you add the template to an application.

Before you begin

Make sure that you created an environment for the application and have resources added to it.

1. Select **Applications > *application name* > Configuration > Webhooks > Add Webhook**.
   The **Add Webhook** window is displayed.
2. Copy the webhook URL and paste into the **URL** field.
3. If you are using a proxy server for Teams, you must specify the following details: **Proxy Host**, **Proxy Port**, **Proxy Username**, and **Proxy Password**.
   These fields can also be automatically populated by setting the corresponding system properties: `proxyHost`, `proxyPort`, `proxyUsername`, and `proxyPassword`.
4. Specify the event type as **Process Failure**.
5. Select the template name that you created in step 3 from the **Webhook Template** field.
6. Specify the environment name in the **Environments** field.
7. Submit your changes.

Results

The webhook template is added to the application. When an application process run on the environment and fails, the Teams channel shows the following example notification.

Webhook-Integration-Test 6/13 6:53 PM
An approval failed for the App application on the Env environment
An application process was rejected during its approval process
Application: App Environment: Env Requested by: admin Scheduled for: Sun Jun 13 18:52:31 IST 2021 Description: ${request.Description}
Click **View Details** for more information.

Teams message card templates

Example templates are provided that helps you to create message cards to use with MS Teams integration.

About this task

Message layouts are available for the following events:

- Application Deployment Failure JSON body on page 2016
- Application Deployment Success JSON body on page 2019
- Application Process Request Started JSON body on page 2022
- Approval Completed JSON body on page 2022
- Approval Failed JSON body on page 2023
- External Approval Completed JSON body on page 2024
- External Approval Failed JSON body on page 2025
- Generic Process Completed JSON body on page 2026
- Generic Process Request Started JSON body on page 2026
- Scheduled Process Not Started JSON body on page 2027

You can customize these available message card as per your team requirement. See MS Teams documentation for more information.

Application Deployment Failure JSON body

```
## PROPERTY Content-Type: application/json

#set ( $Q = '\\"' )

#set ( $themeColor = "#E88686" )
#set ( $warningCount = $workflowTrace.WarningCount)

#macro (showSummary $analysis $level )
  #set ($indentPixels = $level * 20)
  #set ($trace = $analysis.Trace)

  #set ( $text = "${text} <tr>" )
```

```
  #set ( $text = "${text} <td style=${Q}padding-left: ${indentPixels}px;${Q}>$trace.Name</td><td>" )
  #set ($activityResource = $trace.getPropertyValue("activity.resource.id"))
  #if ($activityResource)
    #set ( $text = "${text} $resourceFactory.getResourceForId($fn.getUUID($activityResource)).Path" )
  #else
    #set ( $text = "${text} $trace.getPropertyValue(\"resourcePath\")" )
  #end
  #set ( $text = "${text} </td><td>" )
  #if ($trace.StartedOn)
    #set ( $text = "${text} $fn.formatDate($trace.StartedOn)" )
  #else
    #set ( $text = "${text} Activity not started" )
  #end
  #set ( $text = "${text} </td><td>" )
  #if ($trace.StartedOn && $trace.EndedOn)
    #set ( $text = "${text} $fn.duration($trace.StartedOn, $trace.EndedOn)s" )
  #end
  #set ( $text = "${text} </td><td>$fn.getStatusDisplayName($trace.Result)</td>" )
  #set ( $text = "${text} </tr>" )
  #foreach ($child in $analysis.Children)
    #set ($nextLevel = $level + 1)
    #showSummary ($child, $nextLevel)
  #end
#end

 #macro (showFailure $analysis $level)
   #set ($indentPixels = $level * 20)
   #set ( $text = "${text} <tr background-color=${Q}white${Q}><td style=${Q}padding-left: ${indentPixels}px;
 font-weight: bold; color: #A84F4F${Q}>$analysis.Trace.Name </td></tr>" )
   #if ($analysis.Children.isEmpty())
     #set ($indentPixels = $indentPixels + 20)
     #set ( $text = "${text} <td style=${Q}padding-left: ${indentPixels}px; white-space:
 pre-wrap${Q}>$analysis.FailureMessage</td>" )
   #end
   #foreach ($child in $analysis.Children)
     #set ($nextLevel = $level + 1)
     #showFailure ($child, $nextLevel)
   #end
 #end



#set( $text = "" )
#set( $text = "${text} <h2>A process on $application.Name started</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>$application.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Process:</td><td>$applicationProcess.Name</td></tr>" )
#if ($snapshot)
  #set( $text = " ${text} <tr><td>Snapshot:</td><td>$snapshot.Name</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Environment:</td><td>$environment.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested on:</td><td>$fn.formatDate(${request.SubmittedDate})</td></tr>" )
#if (! $request.Description.isEmpty())
    #set( $text = "${text}   <tr><td>Description:</td><td>${request.Description}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Warning Count:</td><td>${workflowTrace.WarningCount}</td></tr>" )
#if ($workflowTrace.ChildWarningMessages)
```

```
  #set( $text = " ${text} <tr><td>Warnings:</td><td><ul>" )
  #foreach ($message in $workflowTrace.ChildWarningMessages )
    #set ($messageOverLimit = $message.length() > 100)
    #if ($messageOverLimit)
      #set ( $text = "${text} <li>$message.substring(0,100)...</li>" )
    #else
      #set ( $text = "${text} <li>$message</li>" )
    #end
    #set ($remainingWarnings = $warningCount - $workflowTrace.ChildWarningMessages.size())
    #if ($remainingWarnings > 0)
      #set ( $text = "${text} <li>+ $remainingWarnings From Child Workflows</li>" )
    #end
  #end
  #set( $text = " ${text} </ul></td></tr>" )
#end
#set( $text = "${text} </table>" )

#set( $text = " ${text} <div>" )
#set( $text = " ${text} <h3>Process summary:</h3>" )
#set( $text = " ${text} <table cellpadding=${Q}4${Q} cellspacing=${Q}1${Q} width=${Q}100%${Q}>" )
#set( $text = " ${text}  <thead>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Process</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Resource</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Start</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Duration</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Status</th>" )
#set( $text = " ${text}  </thead>" )
#set( $text = " ${text}  <tbody>" )
  #foreach ($child in $analysis.RootAnalyses)
    #showSummary ($child 0)
  #end
#set( $text = " ${text}  </tbody>" )
#set( $text = " ${text} </table>" )
#set( $text = " ${text} </div>" )

#set( $text = " ${text} <div><h3>Failure details:</h3>" )
#set( $text = " ${text}   <table cellpadding=${Q}2${Q} cellspacing=${Q}1${Q} width=${Q}100%${Q}>" )
#set( $text = " ${text}     <table-body>" )
                              #foreach ($child in $failureAnalysis.RootAnalyses)
                                #showFailure ($child 0)
                              #end
#set( $text = " ${text}  </tbody>" )
#set( $text = " ${text} </table>" )
#set( $text = " ${text} </div>" )



#set( $text = " ${text} <div>" )
#if ($versions.size() > 0)
  #set( $text = " ${text} <h3>Included versions</h3>" )
  #set( $text = " ${text} <table cellpadding=${Q}4${Q} cellspacing=${Q}1${Q} width=${Q}100%${Q}>" )
   #set( $text = " ${text} <thead>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Component</th>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Version</th>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Description</th>" )
   #set( $text = " ${text} </thead>" )
   #set( $text = " ${text} <tbody>" )
    #foreach ($version in $versions)
      #set( $text = " ${text} <tr>" )
```

```
       #set( $text = " ${text} <td>$version.Component.Name</td>" )
       #set( $text = " ${text} <td>$version.Name</td>" )
       #set( $text = " ${text} <td>$version.Description</td>" )
     #set( $text = " ${text} </tr>" )
    #end
#set( $text = " ${text}    </tbody>" )
#set( $text = " ${text}   </table>" )
#else
#set( $text = " ${text}   <h3>No versions included</h3>" )
#end
#set( $text = " ${text} </div>" )

#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )



{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "A process on $fn.escapeJson($application.Name) failed",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#applicationProcessRequest/${request.Id}"
                }
            ]
        }
    ]
}
```

Application Deployment Success JSON body

```
## PROPERTY Content-Type: application/json

#set ( $Q = '\\"' )

#set ( $themeColor = "#0072c6" )
#set ( $warningCount = $workflowTrace.WarningCount)

#macro (showSummary $analysis $level )
  #set ($indentPixels = $level * 20)
  #set ($trace = $analysis.Trace)

  #set ( $text = "${text} <tr>" )
  #set ( $text = "${text} <td style=${Q}padding-left: ${indentPixels}px;${Q}>$trace.Name</td><td>" )
  #set ($activityResource = $trace.getPropertyValue("activity.resource.id"))
  #if ($activityResource)
    #set ( $text = "${text} $resourceFactory.getResourceForId($fn.getUUID($activityResource)).Path" )
  #else
    #set ( $text = "${text} $trace.getPropertyValue(\"resourcePath\")" )
  #end
  #set ( $text = "${text} </td><td>" )
```

```
  #if ($trace.StartedOn)
    #set ( $text = "${text} $fn.formatDate($trace.StartedOn)" )
  #else
    #set ( $text = "${text} Activity not started" )
  #end
  #set ( $text = "${text} </td><td>" )
  #if ($trace.StartedOn && $trace.EndedOn)
    #set ( $text = "${text} $fn.duration($trace.StartedOn, $trace.EndedOn)s" )
  #end
  #set ( $text = "${text} </td><td style=${Q}background-color:
 $fn.getStatusColor($trace.Result)${Q}>$fn.getStatusDisplayName($trace.Result)</td>" )
  #set ( $text = "${text} </tr>" )
  #foreach ($child in $analysis.Children)
    #set ($nextLevel = $level + 1)
    #showSummary ($child, $nextLevel)
  #end
#end



#set( $text = "" )
#set( $text = "${text} <h2>A process on $application.Name started</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>$application.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Process:</td><td>$applicationProcess.Name</td></tr>" )
#if ($snapshot)
  #set( $text = " ${text} <tr><td>Snapshot:</td><td>$snapshot.Name</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Environment:</td><td>$environment.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested on:</td><td>$fn.formatDate(${request.SubmittedDate})</td></tr>" )
#if (! $request.Description.isEmpty())
#set( $text = "${text}   <tr><td>Description:</td><td>${request.Description}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Warning Count:</td><td>${workflowTrace.WarningCount}</td></tr>" )
#if ($workflowTrace.ChildWarningMessages)
  #set( $text = " ${text} <tr><td>Warnings:</td><td><ul>" )
  #foreach ($message in $workflowTrace.ChildWarningMessages )
    #set ($messageOverLimit = $message.length() > 100)
    #if ($messageOverLimit)
      #set ( $text = "${text} <li>$message.substring(0,100)...</li>" )
    #else
      #set ( $text = "${text} <li>$message</li>" )
    #end
    #set ($remainingWarnings = $warningCount - $workflowTrace.ChildWarningMessages.size())
    #if ($remainingWarnings > 0)
      #set ( $text = "${text} <li>+ $remainingWarnings From Child Workflows</li>" )
    #end
  #end
  #set( $text = " ${text} </ul></td></tr>" )
#end
#set( $text = "${text} </table>" )

#set( $text = " ${text} <div>" )
#set( $text = " ${text} <h3>Process summary:</h3>" )
#set( $text = " ${text} <table cellpadding=${Q}4${Q} cellspacing=${Q}1${Q} width=${Q}100%${Q}>" )
#set( $text = " ${text}  <thead>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Process</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Resource</th>" )
```

```
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Start</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Duration</th>" )
#set( $text = " ${text}     <th style=${Q}text-align:left;${Q}>Status</th>" )
#set( $text = " ${text}  </thead>" )
#set( $text = " ${text}  <tbody>" )
  #foreach ($child in $analysis.RootAnalyses)
    #showSummary ($child 0)
  #end
#set( $text = " ${text}  </tbody>" )
#set( $text = " ${text} </table>" )
#set( $text = " ${text} </div>" )

#set( $text = " ${text} <div>" )
#if ($versions.size() > 0)
  #set( $text = " ${text} <h3>Included versions</h3>" )
  #set( $text = " ${text} <table cellpadding=${Q}4${Q} cellspacing=${Q}1${Q} width=${Q}100%${Q}>" )
   #set( $text = " ${text} <thead>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Component</th>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Version</th>" )
      #set( $text = " ${text} <th style=${Q}text-align:left;${Q}>Description</th>" )
   #set( $text = " ${text} </thead>" )
   #set( $text = " ${text} <tbody>" )
    #foreach ($version in $versions)
      #set( $text = " ${text} <tr>" )
       #set( $text = " ${text} <td>$version.Component.Name</td>" )
       #set( $text = " ${text} <td>$version.Name</td>" )
       #set( $text = " ${text} <td>$version.Description</td>" )
     #set( $text = " ${text} </tr>" )
    #end
#set( $text = " ${text}    </tbody>" )
#set( $text = " ${text}   </table>" )
#else
#set( $text = " ${text}   <h3>No versions included</h3>" )
#end
#set( $text = " ${text} </div>" )

#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )



{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "A process on $fn.escapeJson($application.Name) completed successfully",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#applicationProcessRequest/${request.Id}"
                }
            ]
        }
    ]
}
```

Application Process Request Started JSON body

```
## PROPERTY Content-Type: application/json

#set ( $themeColor = "#0072c6" )

#set( $text = "" )
#set( $text = "${text} <h2>A process on $application.Name started</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>$application.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Process:</td><td>$applicationProcess.Name</td></tr>" )
#if ($snapshot)
  #set( $text = " ${text} <tr><td>Snapshot:</td><td>$snapshot.Name</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Environment:</td><td>$environment.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested on:</td><td>$fn.formatDate(${request.SubmittedDate})</td></tr>" )
#set( $text = "${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "A process on $fn.escapeJson($application.Name) started",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#applicationProcessRequest/${request.Id}"
                }
            ]
        }
    ]
}
```

Approval Completed JSON body

```
## PROPERTY Content-Type: application/json



#set ( $themeColor = "#0072C6" )
#if ( $appRequest )
    #set ( $processName = $appRequest.getApplicationProcess().getName() )
#end
#if ( $compRequest )
    #set ( $processName = $compRequest.getComponentProcess().getName() )
#end

#set( $text = "" )
#set( $text = "${text} <h2>HCL DevOps Deploy (Deploy) Process: $processName</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>${application.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Environment:</td><td>${environment.Name}</td></tr>" )
```

```
#if ($snapshot)
  #set( $text = "${text} <tr><td>Snapshot:</td><td>${snapshot.Name}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Requested by:</td><td>${requester.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Scheduled for:</td><td>${scheduledDate}</td></tr>" )
#if (!$appRequest.getDescription().isEmpty())
#set( $text = "${text}   <tr><td>Description:</td><td>${appRequest.getDescription()}</td></tr>" )
#end
#set( $text = "${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "An approval was completed for the $fn.escapeJson($application.Name) application on the
 $fn.escapeJson($environment.Name) environment",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
    ]
}
```

Approval Failed JSON body

```
## PROPERTY Content-Type: application/json

#set( $themeColor = "#E88686" )
#if ( $appRequest )
    #set ( $processName = $appRequest.getApplicationProcess().getName() )
#end
#if ( $compRequest )
    #set ( $processName = $compRequest.getComponentProcess().getName() )
#end

#set( $text = "" )
#set( $text = "${text} <h2>HCL DevOps Deploy (Deploy) Process $processName was rejected during its approval
 process</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>${application.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Environment:</td><td>${environment.Name}</td></tr>" )
#if ($snapshot)
  #set( $text = "${text} <tr><td>Snapshot:</td><td>${snapshot.Name}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Requested by:</td><td>${requester.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Scheduled for:</td><td>${scheduledDate}</td></tr>" )
#if (!$appRequest.getDescription().isEmpty())
#set( $text = "${text}   <tr><td>Description:</td><td>${appRequest.getDescription()}</td></tr>" )
#end
#set( $text = "${text} </table>" )
```

```
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "An approval failed for the $fn.escapeJson($application.Name) application on the
 $fn.escapeJson($environment.Name) environment",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
    ]
}
```

External Approval Completed JSON body

```
## PROPERTY Content-Type: application/json

#set( $themeColor = "#0072C6")
#if ( $appProcRequest )
    #set ( $description = $appProcRequest.getDescription() )
#end
#if ( $compProcRequest )
    #set ( $description = $compProcRequest.getDescription() )
#end

#set( $text = "" )
#set( $text = "${text} <h2>An external approval process was successful</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>${application.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Environment:</td><td>${environment.Name}</td></tr>" )
#if ($snapshot)
  #set( $text = "${text} <tr><td>Snapshot:</td><td>${snapshot.Name}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Requested by:</td><td>${requester.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Scheduled for:</td><td>${scheduledDate}</td></tr>" )
#if ($description && !$description.isEmpty())
#set( $text = "${text}   <tr><td>Description:</td><td>${description}</td></tr>" )
#end
#set( $text = "${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "An external approval was completed for the $fn.escapeJson($application.Name) application on the
 $fn.escapeJson($environment.Name) environment",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
```

```
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
    ]
}
```

External Approval Failed JSON body

```
## PROPERTY Content-Type: application/json

#set( $themeColor = "#E88686" )
#if ( $appProcRequest )
    #set ( $description = $appProcRequest.getDescription() )
#end
#if ( $compProcRequest )
    #set ( $description = $compProcRequest.getDescription() )
#end

#set( $text = "" )
#set( $text = "${text} <h2>An application process was rejected during its external approval process</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Application:</td><td>${application.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Environment:</td><td>${environment.Name}</td></tr>" )
#if ($snapshot)
  #set( $text = "${text} <tr><td>Snapshot:</td><td>${snapshot.Name}</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Requested by:</td><td>${requester.Name}</td></tr>" )
#set( $text = "${text}   <tr><td>Scheduled for:</td><td>${scheduledDate}</td></tr>" )
#if ($description && !$description.isEmpty())
#set( $text = "${text}   <tr><td>Description:</td><td>${description}</td></tr>" )
#end
#set( $text = "${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "An external approval failed for the $fn.escapeJson($application.Name) application on the
 $fn.escapeJson($environment.Name) environment",
    "text": "$fn.escapeJson($text)",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
```

```
    ]
}
```

Generic Process Completed JSON body

```
## PROPERTY Content-Type: application/json

#set( $themeColor = $fn.getStatusColor( $request.Result ) )
#if ( $themeColor.contains("#FFFFFF") )
  #set ( $themeColor = "#0072C6" )
#end

#set( $text = "" )
#set( $text = "${text} <h2>HCL DevOps Deploy (Deploy) Process $process.Name
 $fn.getStatusDisplayName($request.Result)</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Process:</td><td>$process.Name</td></tr>" )
#if ($resourceName)
  #set( $text = " ${text} <tr><td>Resource:</td><td>$resourceName</td></tr>" )
#end
#set( $text = "${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = "${text}   <tr><td>Requested on:</td><td>$scheduledDate</td></tr>" )
#set( $text = "${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information" )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "HCL DevOps Deploy (Deploy) Process $fn.escapeJson($process.Name)
 $fn.getStatusDisplayName($request.Result)",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
    ]
}
```

Generic Process Request Started JSON body

```
## PROPERTY Content-Type: application/json

#set ( $themeColor = "#0072c6" )

#set( $text = "" )
#set( $text = "${text} <h2>HCL DevOps Deploy (Deploy) Process $process.Name started</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Process:</td><td>$process.Name</td></tr>" )
#if ($resourceName)
  #set( $text = " ${text} <tr><td>Resource:</td><td>$resourceName</td></tr>" )
```

```
#end
#set( $text = " ${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = " ${text}   <tr><td>Requested on:</td><td>$scheduledDate</td></tr>" )
#set( $text = " ${text} </table>" )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information" )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "HCL DevOps Deploy (Deploy) Process $fn.escapeJson($process.Name) started",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
                }
            ]
        }
    ]
}
```

Scheduled Process Not Started JSON body

```
## PROPERTY Content-Type: application/json

#set ( $themeColor = "#E88686" )

#set( $text = "" )
#set( $text = "${text} <h2>HCL DevOps Deploy (Deploy) Process $process.Name failed to start</h2>" )
#set( $text = "${text} <table> " )
#set( $text = "${text}   <tr><td>Process:</td><td>$process.Name</td></tr>" )
#if ($resourceName)
  #set( $text = " ${text} <tr><td>Resource:</td><td>$resourceName</td></tr>" )
#end
#set( $text = " ${text}   <tr><td>Requested by:</td><td>$user.Name</td></tr>" )
#set( $text = " ${text}   <tr><td>Requested on:</td><td>$scheduledDate</td></tr>" )
#set( $text = " ${text} </table>" )
#set( $text = " ${text} <br>The process failed to start because the server is in maintenance mode. You can
 request the process again when maintenance is complete." )
#set( $text = "${text} <br>Click <strong>View Details</strong> for more information." )
{
    "@context": "https://schema.org/extensions",
    "@type": "MessageCard",
    "themeColor": "$themeColor",
    "title": "HCL DevOps Deploy (Deploy) Process $fn.escapeJson($process.Name) failed to start",
    "text": "$fn.escapeJson($text).trim()",
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "View Details",
            "targets": [
                {
                    "os": "default",
                    "uri": "$fn.escapeJson($externalUrl)#$requestType/$processId"
```

```
                }
            ]
        }
    ]
}
```

How-to: Integrate Slack

Integrate Slack with HCL DevOps Deploy (Deploy) to post notifications automatically to your Slack app whenever a specific deployment event happens.

About this task

Walk through a scenario and learn how to push messages to Slack when an application process fails to deploy in Deploy.

In this example, you'll learn about the requirements and procedure to integrate Slack with Deploy. To integrate Slack with Deploy, follow the steps provided in this section:

Step 1: Create an incoming webhook app in Slack

1. Go to api.slack.com and click **Create an App**.
2. Provide an App Name and choose an appropriate workspace.
3. After you create the app, go to **Incoming Webhooks**.
4. Turn on **Activate Incoming Webhooks**, and click **Add New Webhook to Workspace**.
5. Choose a Slack channel that you want the post the messages to.
6. Click **Allow**.
7. Copy and save the generated webhook URL.

Results

The webhook is added to the Slack app and posts a confirmation message in the app.

Note: For additional information about incoming webhooks for Slack, refer to the Slack documentation.

Step 2: Define the message layout

A message layout is defined in a JSON format that includes the message in text, links, and image, if required. For HCL DevOps Deploy (Deploy), you can create message layout that refers to the process name, status, and a link to the process deployment page. This message layout is used in the webhook template. Message layout is defined differently for each event type.

About this task

The message layout is available for the following events:

- Application Deployment Failure JSON body on page 2030
- Application Deployment Success JSON body on page 2034
- Approval Completed JSON body on page 2037
- Approval Created JSON body on page 2039

- Approval Failed JSON body on page 2042
- External Approval Completed JSON body on page 2044
- External Approval Failed JSON body on page 2045
- Generic Process Failure JSON body on page 2046
- Generic Process Request Started JSON body on page 2047
- Generic Process Success JSON body on page 2047
- Process Not Started JSON body on page 2048
- Process Request Started JSON body on page 2049
- Task Created JSON body on page 2052

Step 3: Create a webhook template

Create a webhook template for an application process failure so that when the event happens a notification is triggered in your Slack app.

1. Select **Settings > Webhook Templates > Create Webhook Template**.
2. Provide a name and description for the template. For this example, give the name as `Application process failed notification`.
3. Click on **Body** and paste the example Application Deployment Failure JSON body from the step 2.
4. Choose the event type **Process Failure** from the **Type** field.
5. Submit your changes.

Results

The webhook template is created and ready to be added to an application.

Step 4: Add the webhook template to an application

You can add a webhook template to applications, generic processes, and external approval processes. For this example, you add the template to an application.

Before you begin

Make sure that you have created an environment for the application and have resources added to it.

1. Select **Applications > *application name* > Configuration > Webhooks > Add Webhook**.
   The **Add Webhook** window is displayed.
2. Copy the URL that you generated from step 1 and paste it in the **URL** field.
3. Specify the event type as **Process Failure**.
4. Select the template name created in step 3 from the **Webhook Template** field.
5. Specify the environment name from the **Environments** field.
6. Submit your changes.

Results

The webhook template is added to the application. When the application process runs on the environment and fails, the Slack app shows the following example notification.

Webhook-Integration-Test 6/13 6:53 PM

An approval failed for the App application on the Env environment

An application process was rejected during its approval process

Application: App Environment: Env Requested by: admin Scheduled for: Sun Jun 13 18:52:31 IST 2021 Description: ${request.Description}

Click **View Details** for more information.

Slack message layout templates

Example templates are provided that helps you to create message layout to use with Slack integration.

About this task

The message layout is available for the following events:

- Application Deployment Failure JSON body on page 2030
- Application Deployment Success JSON body on page 2034
- Approval Completed JSON body on page 2037
- Approval Created JSON body on page 2039
- Approval Failed JSON body on page 2042
- External Approval Completed JSON body on page 2044
- External Approval Failed JSON body on page 2045
- Generic Process Failure JSON body on page 2046
- Generic Process Request Started JSON body on page 2047
- Generic Process Success JSON body on page 2047
- Process Not Started JSON body on page 2048
- Process Request Started JSON body on page 2049
- Task Created JSON body on page 2052

Note: You can customize these available message layout as per your team requirement. Refer to Slack documentation for more information.

Application Deployment Failure JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
#macro (addLimitReached)
        ## show limit message & link
    ,{
        "type": "section",
        "text": {
            "type": "mrkdwn",
            "text": "<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View
 Process Request For More>"
```

```
        }
    }
        #set ($blockCount = $blockCount + 1)
#end
#macro (showFailure $analysis $indent)
    #if ($blockCount == 50)
        #addLimitReached()
    #elseif ($blockCount > 50)
        ##ignore no room left
    #else
        ## add section and recurse down
    ,{
        "type": "section",
        "text": {
            "type": "plain_text",
            "text": "$indent$fn.escapeJson($analysis.getTrace().getName())"
        }
    }
        #set ($blockCount = $blockCount + 1)
        #set ($analysisChildren = $analysis.getChildren())
        #if ($analysisChildren && $analysisChildren.size() > 0)
            #set ($newindent = $indent+"--")
            #foreach ($child in $analysisChildren)
                #showFailure($child, $newindent)
            #end
        #end
    #end
#end

#macro (showVersion $version)
    #if ($blockCount == 50)
        #addLimitReached()
    #elseif ($blockCount > 50)
        ##ignore no room left
    #else
        ## add section
    ,{
        "type": "section",
        "fields": [{
             "type": "mrkdwn",
             "text": "*Component*: $fn.escapeJson($version.getComponent().getName())"
         },{
             "type": "mrkdwn",
             "text": "*Version*: $fn.escapeJson($version.getName())"
         #if ($version.getDescription() and !$version.getDescription().trim().isEmpty())
         },{
             "type": "plain_text",
             "text": "$fn.escapeJson($version.getDescription())"
         #end
         }]
    }
        #set ($blockCount = $blockCount + 1)
    #end
#end

{
    "blocks": [
        ## block 1
```

```
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process on $fn.escapeJson($application.getName()) failed"
            }
        },
        ## block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($applicationProcess.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Requested On*: $fn.formatDate($request.getSubmittedDate())"
                }
                ,{
                    "type": "mrkdwn",
                    "text": "*Duration in Milliseconds*: $request.getDurationInMillis()"
                }
                #if ($request.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Description*: $fn.escapeJson($request.getDescription())"
                }
                #end
                #set ($warningCount = $workflowTrace.getWarningCount())
                #if ($warningCount > 0)
                ,{
                    "type": "mrkdwn",
                    "text": "*Warning Count*: $warningCount"
                }
                #end
            ]
        },
        ## block 3
        {
```

```
            "type":"section",
            "text": {
                "type":"mrkdwn",
                "text":
"<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View Process Request>"
            }
        },
        ## block 4
        {
            "type":"divider"
        },
        ## block 5
        #set ($warnMessList = $workflowTrace.getChildWarningMessages())
        #if ($warnMessList && $warnMessList.size() > 0)
        {
            "type": "section",
            "fields": [{
                "type": "mrkdwn",
                "text": "*Warnings*"
            #foreach ($mess in $warnMessList)
                #if($velocityCount == 9)
                    ## add link
            },{
                "type": "mrkdwn",
                "text":
"<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|See process request for
more>"
                #elseif ($velocityCount > 9)
                    ## ignore
                #else
            },{
                "type": "plain_text",
                "text": "$fn.escapeJson($mess)"
                #end
            #end
            }]
        },
        #end
        ## block 6
        {
            "type":"divider"
        }
        ## Show as many of the failure analysis and versions as we can.
        ## If we run out of blocks(slack limits to 50), make the last block
        ## a link to the process request
        ## for brevity we don't show summary of other steps as we do in emails
        ##block 7 - 50
        #set ($blockCount = 7)
        #set ($rootAnalyses = $failureAnalysis.getRootAnalyses())
        #if ($rootAnalyses && $rootAnalyses.size() > 0)
            ## block 7
            ,{
                "type": "section",
                "text": {
                    "type": "mrkdwn",
                    "text": "*Failures*"
                }
            }
```

```
            #set ($blockCount = $blockCount + 1)
            #foreach ($failure in $rootAnalyses)
                #showFailure ($failure "")
            #end
        #end
        ##Show as many versions as we have space for
        #if ($versions && $versions.size() > 0)
            #if ($blockCount == 50)
                #addLimitReached()
            #elseif ($blockCount > 50)
                ## do nothing
            #else
                ,{
                    "type": "section",
                    "text": {
                        "type": "mrkdwn",
                        "text": "*Versions*"
                    }
                }
                #set ($blockCount = $blockCount + 1)
                #foreach($version in $versions)
                    #showVersion ($version)
                #end
            #end
        #end
    ]
}
## END SECTION Body
```

Application Deployment Success JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
#macro (addLimitReached)
        ## show limit message & link
    ,{
        "type": "section",
        "text": {
            "type": "mrkdwn",
            "text": "<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View
 Process Request For More>"
        }
    }
        #set ($blockCount = $blockCount + 1)
#end

#macro (showVersion $version)
    #if ($blockCount == 50)
        #addLimitReached()
    #elseif ($blockCount > 50)
        ##ignore no room left
    #else
        ## add section
    ,{
        "type": "section",
        "fields": [{
             "type": "mrkdwn",
             "text": "*Component*: $fn.escapeJson($version.getComponent().getName())"
```

```
         },{
             "type": "mrkdwn",
             "text": "*Version*: $version.getName()"
         #if ($version.getDescription() and !$version.getDescription().trim().isEmpty())
         },{
             "type": "plain_text",
             "text": "$fn.escapeJson($version.getDescription())"
         #end
         }]
    }
        #set ($blockCount = $blockCount + 1)
    #end
#end

{
    "blocks": [
        ## block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process on $fn.escapeJson($application.getName()) completed successfully"
            }
        },
        ## block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($applicationProcess.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Requested On*: $fn.formatDate($request.getSubmittedDate())"
                }
                ,{
                    "type": "mrkdwn",
                    "text": "*Duration in Milliseconds*: $request.getDurationInMillis()"
                }
```

```
                #if ($request.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Description*: $fn.escapeJson($request.getDescription())"
                }
                #end
                #set ($warningCount = $workflowTrace.getWarningCount())
                #if ($warningCount > 0)
                ,{
                    "type": "mrkdwn",
                    "text": "*Warning Count*: $warningCount"
                }
                #end
            ]
        },
        ## block 3
        {
            "type":"section",
            "text": {
                "type":"mrkdwn",
                "text":
"<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View Process Request>"
            }
        },
        ## block 4
        {
            "type":"divider"
        },
        ## block 5
        #set ($warnMessList = $workflowTrace.getChildWarningMessages())
        #if ($warnMessList && $warnMessList.size() > 0)
        {
            "type": "section",
            "fields": [{
                "type": "mrkdwn",
                "text": "*Warnings*"
            #foreach ($mess in $warnMessList)
                #if($velocityCount == 9)
                    ## add link
            },{
                "type": "mrkdwn",
                "text":
"<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|See process request for
more>"
                #elseif ($velocityCount > 9)
                    ## ignore
                #else
            },{
                "type": "plain_text",
                "text": "$fn.escapeJson($mess)"
                #end
            #end
            }]
        },
        #end
        ## block 6
        {
            "type":"divider"
```

```
        }
        ##block 7 - 50
        #set ($blockCount = 7)
        ## show as many versions as possible
        ## in the email template we show step summaries as well.
        ## this is omitted here due to slack limits on sections and
        ## my personal believe version list is more valuable than a bunch
        ## of successful steps
        ##Show as many versions as we have space for
        #if ($versions && $versions.size() > 0)
            #if ($blockCount == 50)
                #addLimitReached()
            #elseif ($blockCount > 50)
                ## do nothing
            #else
                ,{
                    "type": "section",
                    "text": {
                        "type": "mrkdwn",
                        "text": "*Versions*"
                    }
                }
                #set ($blockCount = $blockCount + 1)
                #foreach($version in $versions)
                    #showVersion ($version)
                #end
            #end
        #end
    ]
}
## END SECTION Body
```

Approval Completed JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "Approval Completed"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
```

```
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($requester.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Scheduled for*: $scheduledDate"
                }
                #if ($appRequest.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($appRequest.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        },
        ## Block 4
        {
            "type": "divider"
        }
        ## Block 5
        #if ($tasks && $tasks.size() > 0)
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "*Tasks*"
            }
        }
        #end
        #set ($blockCount = 6)

        #foreach ($task in $tasks)
            #if ($blockCount == 50)
        ,{
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request for more
information.>"
            }
        }
            #set ($blockCount = $blockCount + 1)
            #elseif ($blockCount > 50)
```

```
                ## no more room
            #else
        ,{
            "type": "context",
            "elements": [{
                "type": "mrkdwn",
                "text": "*Name*: $fn.escapeJson($task.getName())"
            },{
                "type": "mrkdwn",
                "text": "*Type*: $fn.escapeJson($task.getTaskType())"
            },{
                "type": "mrkdwn",
                "text": "*Status*: $fn.escapeJson($task.getStatus().name())"
            },{
                "type": "mrkdwn",
                "text": "*User*: $fn.escapeJson($task.getUser().getDisplayName())"
            },{
                "type": "mrkdwn",
                "text": "*Comment*: $fn.escapeJson($task.getComment())"
            },{
                "type": "mrkdwn",
                "text": "*Date*: $fn.formatUnixTime($task.getDateEnded())"
            }]
        }
            #set ($blockCount = $blockCount + 1)
            #end
        #end
    ]
}
## END SECTION Body
```

Approval Created JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "Approval Pending"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
```

```
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                #if ($component)
                {
                    "type": "mrkdwn",
                    "text": "*Component*: $fn.escapeJson($component.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($processName)"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Scheduled for*: $scheduledDate"
                }
                #if ($appRequest.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($appRequest.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        },
        ## Block 4
        #if ($respondUrl)
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($respondUrl)|Respond>"
            }
        },
        #end
        ## Block 5
        {
            "type": "divider"
        }
        ## Block 6
        #if ($roles)
        ,{
            "type": "section",
            "fields": [
```

```
                {
                    "type": "mrkdwn",
                    "text": "*Roles*:"
                }
                ## This breaks at > 9 roles. Check for version handling to modify to handle more
                #foreach ( $role in $roles )
                ,{
                     "type": "mrkdwn",
                     "text": "$fn.escapeJson($role.getName())"
                }
                #end
            ]
        },{
            "type": "divider"
        }
        #end
        ## Block 7-50
        #if ($versionMap)
            #set ( $compSet = $versionMap.getComponents())
            #foreach ( $comp in $compSet)
                ## velocityCount is the default iterator velocity uses
                #if ($velocityCount == 42)
                    ## add placeholder
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|Refer to request for more
components.>"
            }
        }
                #elseif ($velocityCount > 42)
                     ## do nothing
                     ## anymore and slack will reject the message
                ## velocityCount starts at 1
                #elseif ($velocityCount == 1)
                    ## add label and thing
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "*Versions by Component*:"
            }
        }
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "*$fn.escapeJson($comp)*: $fn.escapeJson($versionMap.getVersionsForComp($comp))"
            }
        }
                #else
                     ## addthing
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "*$fn.escapeJson($comp)*: $fn.escapeJson($versionMap.getVersionsForComp($comp))"
```

```
            }
        }
                #end
            #end
        #end
    ]
}
## END SECTION Body
```

Approval Failed JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "Approval Failed"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($requester.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Scheduled for*: $scheduledDate"
                }
                #if ($appRequest.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($appRequest.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
```

```
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        },
        ## Block 4
        {
            "type": "divider"
        }
        ## Block 5
        #if ($tasks && $tasks.size() > 0)
        ,{
            "type": "section",
            "text": {
                "type": "mrkdwn",
                "text": "*Tasks*"
            }
        }
        #end
        #set ($blockCount = 6)

        #foreach ($task in $tasks)
            #if ($blockCount == 50)
        ,{
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request for more
information.>"
            }
        }
            #set ($blockCount = $blockCount + 1)
            #elseif ($blockCount > 50)
                ## no more room
            #else
        ,{
            "type": "context",
            "elements": [{
                "type": "mrkdwn",
                "text": "*Name*: $fn.escapeJson($task.getName())"
            },{
                "type": "mrkdwn",
                "text": "*Type*: $fn.escapeJson($task.getTaskType())"
            },{
                "type": "mrkdwn",
                "text": "*Status*: $$fn.escapeJson(task.getStatus().name())"
            },{
                "type": "mrkdwn",
                "text": "*User*: $fn.escapeJson($task.getUser().getDisplayName())"
            },{
                "type": "mrkdwn",
                "text": "*Comment*: $fn.escapeJson($task.getComment())"
            },{
                "type": "mrkdwn",
                "text": "*Date*: $fn.formatUnixTime($task.getDateEnded())"
            }]
```

```
        }
            #set ($blockCount = $blockCount + 1)
            #end
        #end
    ]
}
## END SECTION Body
```

External Approval Completed JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "An external approval process was successful"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($requester.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Scheduled for*: $scheduledDate"
                }
                #if ($request.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($request.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
        {
```

```
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
}
## END SECTION Body
```

External Approval Failed JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "An application process was rejected during its external approval process"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                },
                #if ($snapshot)
                {
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },
                #end
                {
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($requester.getDisplayName())"
                },
                {
                    "type": "mrkdwn",
                    "text": "*Scheduled for*: $scheduledDate"
                }
                #if ($request.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($request.getDescription())"
                }
                #end
            ]
        },
```

```
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
}
## END SECTION Body
```

Generic Process Failure JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process failed"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($process.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $scheduledDate"
                }
                #if ($resourceName)
                ,{
                    "type": "mrkdwn",
                    "text": "*Resource*: $fn.escapeJson($resourceName)"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
```

```
}
## END SECTION Body
```

Generic Process Request Started JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process started"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($process.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $scheduledDate"
                }
                #if ($resourceName)
                ,{
                    "type": "mrkdwn",
                    "text": "*Resource*: $fn.escapeJson($resourceName)"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
}
## END SECTION Body
```

Generic Process Success JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
```

```
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process succeeded"
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($process.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $scheduledDate"
                }
                #if ($resourceName)
                ,{
                    "type": "mrkdwn",
                    "text": "*Resource*: $fn.escapeJson($resourceName)"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
}
## END SECTION Body
```

Process Not Started JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A process failed to start (server in maintenance mode)"
            }
        },
        ## Block 2
        {
            "type": "context",
```

```
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($process.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $scheduledDate"
                }
                #if ($resourceName)
                ,{
                    "type": "mrkdwn",
                    "text": "*Resource*: $fn.escapeJson($resourceName)"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type": "plain_text",
                "text": "The process failed to start because the server is in maintenance mode. You can request
 the process again when maintenance is complete."
            }
        },
        ## Block 4
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
            }
        }
    ]
}
## END SECTION Body
```

Process Request Started JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
#macro (addLimitReached)
        ## show limit message & link
    ,{
        "type": "section",
        "text": {
            "type": "mrkdwn",
            "text": "<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View
 Process Request For More>"
        }
    }
        #set ($blockCount = $blockCount + 1)
#end

#macro (showVersion $version)
```

```
    #if ($blockCount == 50)
        #addLimitReached()
    #elseif ($blockCount > 50)
        ##ignore no room left
    #else
        ## add section
    ,{
        "type": "section",
        "fields": [{
             "type": "mrkdwn",
             "text": "*Component*: $fn.escapeJson($version.getComponent().getName())"
         },{
             "type": "mrkdwn",
             "text": "*Version*: $fn.escapeJson($version.getName())"
         #if ($version.getDescription() and !$version.getDescription().trim().isEmpty())
         },{
             "type": "plain_text",
             "text": "$fn.escapeJson($version.getDescription())"
         #end
         }]
    }
        #set ($blockCount = $blockCount + 1)
    #end
#end
#macro (showChanges $compName)
    #if ($blockCount == 50)
        #addLimitReached()
        #set ($blockCount = $blockCount + 1)
    #elseif ($blockCount > 50)
        ## do nothing
    #else
        ,{
            "type":"section",
            "fields": [
                {
                    "type":"mrkdwn",
                    "text": "*$fn.escapeJson($compName)*:"
                }
                #set ($changes = $changesByComponent.get($compName))
                #foreach ($change in $changes)
                    #if ($velocityCount < 9)
                ,{
                    "type": "mrkdwn",
                    "text": "$fn.escapeJson($fn.getChangeTypeDisplayName($change))
 $fn.escapeJson($fn.getChangeVersionDisplayName($change)) to
 <
$fn.escapeJson($externalUrl)#resource/$change.getResource().getId().toString()|$fn.escapeJson($change.getResour
ce().getName())>"
                }
                    #elseif ($velocityCount == 9)
                ,{
                    "type": "mrkdwn",
                    "text":
 "<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View Process Request For
 More>"
                }
                    #end
                #end
```

```
            ]
        }
        #set ($blockCount = $blockCount + 1)
    #end
#end

{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "An application process started."
            }
        },
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($applicationProcess.getName())"
                },{
                #if ($snapshot)
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                },{
                #end
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $fn.formatDate($request.getSubmittedDate())"
                }
                #if ($request.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Description*: $fn.escapeJson($request.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "text": {
                "type":"mrkdwn",
                "text":
 "<$fn.escapeJson($externalUrl)#applicationProcessRequest/$request.getId().toString()|View Process Request For
 More>"
            }
        }
        #set ($blockCount = 4)
        ##Show as many versions as we have space for
```

```
        #if ($versions && $versions.size() > 0)
            #if ($blockCount == 50)
                #addLimitReached()
            #elseif ($blockCount > 50)
                ## do nothing
            #else
                ,{
                    "type": "section",
                    "text": {
                        "type": "mrkdwn",
                        "text": "*Versions*"
                    }
                }
                #set ($blockCount = $blockCount + 1)
                #foreach($version in $versions)
                    #showVersion ($version)
                #end
            #end
        #end
        ## show as many Components as possible
        #if ($uniqueComponentNames && $uniqueComponentNames.size() > 0)
            #if ($blockCount == 50)
                #addLimitReached()
            #elseif ($blockCount > 50)
                ## do nothing
            #else
                ,{
                    "type": "section",
                    "text": {
                        "type": "mrkdwn",
                        "text": "*Pending Changes*"
                    }
                }
                #set ($blockCount = $blockCount + 1)
                #foreach($compName in $uniqueComponentNames)
                    #showChanges ($compName)
                #end
            #end
        #end
    ]
}
## END SECTION Body
```

Task Created JSON body

```
## BEGIN SECTION Body
## PROPERTY Content-Type: application/json
{
    "blocks": [
        ## Block 1
        {
            "type": "header",
            "text": {
                "type": "plain_text",
                "text": "A task is eligible to be completed. Please review the request and approve or reject
 this task."
            }
        },
```

```
        ## Block 2
        {
            "type": "context",
            "elements": [
                {
                    "type": "mrkdwn",
                    "text": "*Process*: $fn.escapeJson($process.getName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Requested by*: $fn.escapeJson($user.getDisplayName())"
                },{
                    "type": "mrkdwn",
                    "text": "*Request on*: $scheduledDate"
                }
                #if ($application)
                ,{
                    "type": "mrkdwn",
                    "text": "*Application*: $fn.escapeJson($application.getName())"
                }
                #end
                #if ($enviornment)
                ,{
                    "type": "mrkdwn",
                    "text": "*Environment*: $fn.escapeJson($environment.getName())"
                }
                #end
                #if ($snapshot)
                ,{
                    "type": "mrkdwn",
                    "text": "*Snapshot*: $fn.escapeJson($snapshot.getName())"
                }
                #end
                #if ($component)
                ,{
                    "type": "mrkdwn",
                    "text": "*Component*: $fn.escapeJson($component.getName())"
                }
                #end
                #if ($resourceName)
                ,{
                    "type": "mrkdwn",
                    "text": "*Resource*: $fn.escapeJson($resourceName)"
                }
                #end
                #if ($appRequest && $appRequest.getDescription())
                ,{
                    "type": "mrkdwn",
                    "text": "*Request Description*: $fn.escapeJson($appRequest.getDescription())"
                }
                #end
            ]
        },
        ## Block 3
        {
            "type": "section",
            "fields": [
                {
                    "type":"mrkdwn",
```

```
                    "text": "<$fn.escapeJson($externalUrl)/#$requestType/$processId|View Request>"
                }
                #if ($respondUrl)
                ,{
                    "type":"mrkdwn",
                    "text": "<$fn.escapeJson($respondUrl)|Respond>"
                }
                #end
            ]
        }
        #if ($roles)
        ,{
            "type": "section",
            "fields": [
                {
                    "type": "mrkdwn",
                    "text": "*Roles*:"
                }
                ## This breaks at > 9 roles. Check for other templates handling to modify to handle more
                #for each ( $role in $roles )
                ,{
                     "type": "mrkdwn",
                     "text": "$fn.escapeJson($role.getName())"
                }
                #end
            ]
        }
        #end
    ]
}
## END SECTION Body
```

Video and learning resources

Many resources about effectively using HCL DevOps Deploy (Deploy) are available. You can choose which learning resources to use depending on your role, skill level, interests, and the time you have.

The following types of learning resources are available. To learn more about them, including where to find them and what you gain by using them, see the detailed descriptions.

- Videos and multimedia on page 2054
- Tours and tutorials on page 2055
- Learning circles on page 2055
- Additional resources on page 2055

Videos and multimedia

Many videos and multimedia are available to help you learn about a wide range of Deploy capabilities.

- For videos that are related to Deploy, see the following external web links:
  - Deploy video gallery
  - Deploy YouTube playlist
- For videos that are related to HCLSoftware DevOps, see YouTube playlist.

Tours and tutorials

Tutorials provide instructions for common tasks. Some tutorials also explain the concepts that you need to understand before you start a task. Complete a tutorial to learn to do a task, or revisit a tutorial for a reminder of how to do a task that you do not often do. You can access tutorials from Tutorials on page 1914.

Learning circles

A learning circle is a community space where people come to learn about a subject. Learning circles offer targeted resources and guided learning activities that are selected by subject matter experts. Use learning circles for these purposes:

- Building personal learning roadmaps that are based on learning roadmaps that were created by experts
- Discussing learning activities with other learning circle members so that you can deepen your understanding or help others
- Asking questions or joining a conversation in the community forum, where you can find answers to your getting-started questions
- Blogging about your ideas and experiences, and learning from others' blog posts
- Finding experts and more resources

Additional resources

- Much more information is available on the Deploy community.

Installing an external agent for Deploy in DevOps Loop

You can install and connect an external Deploy agent to a DevOps Loop instance running on Kubernetes (K8s), and IBM Kubernetes Service (IKS).

Before you begin

Ensure that the following prerequisites are met:

- Installed DevOps Loop on Kubernetes (K8s), or IKS.
- You have access to the Kubernetes cluster where DevOps Loop is running.

1. Obtain the Deploy web agent communication URL for the Deploy server running in the DevOps Loop instance.
   In a Kubernetes cluster, the URL is indicated as: `wss://service-{{.Values.global.domain}}:7919`

   Here, the value for the `Values.global.domain` is the same URL used to access the DevOps Loop UI.
2. Install the Deploy agent using the Deploy web agent communication URL obtained in step 1.
   During the agent installation, you must specify any teams you want the Deploy agent to be included in. Refer to Installing agents for more information on Deploy agent installation.
3. Start the external Deploy agent.
   Result
   The agent is displayed under the **Agents** section in the Deploy UI.

Troubleshooting

If you encounter issues connecting the Deploy agent to the Deploy server running in DevOps Loop, verify that the Emissary Ingress service is configured to route traffic on port 7919 to the Deploy server WSS service:

a. Run the following command to inspect the service:

```
$ kubectl get svc -n emissary
```

A correct output lists 7919 in PORT(S) column, like the following:

```
NAME                    TYPE         CLUSTER-IP     EXTERNAL-IP PORT(S)
                        AGE
emissary-ingress        LoadBalancer 172.21.20.31   xx.xx.xx.xx
 443:31734/TCP,80:32086/TCP,7919:31941/TCP,9022:31708/TCP 112d
emissary-ingress-admin  ClusterIP    172.21.9.14    <none>      8877/TCP,8005/TCP
                        112d
```

b. If the port 7919 is not listed for the emissary-ingress service, edit the service configuration:

```
$ kubectl edit svc emissary-ingress -n emissary
```

c. Under the `spec.ports` section, add the following lines:

```
- name: deploy-wss
 port: 7919
 protocol: TCP
 targetPort: 7919
```

After port 7919 is configured in the emissary-ingress service, your Deploy agent should successfully connect to the Deploy server running in DevOps Loop.

4. Perform the following steps to manually assign a license type:
   a. Navigate to **Deploy**
   b. Go to **Resources → Agents**
   c. Locate the connected agent
   d. Click the **Actions** menu for the agent
   e. Select **Set License Type**
   f. Choose the appropriate license type:
      - **Authorized (Assigned)** or
      - **Floating**
   g. Click **Save**
5. For IKS, configure the cluster service domain, if it is not configured.
   Follow steps in the **Installing DevOps Loop on IBM Cloud Kubernetes Service (IKS)** topic for setting up the service domain.

Results

You have installed and connected the external Deploy agent to DevOps Loop.

What to do next

You can now use this agent to run deployments.

Overview of integrations

HCL DevOps Deploy (Deploy) supports integrations with other IBM® and HCL products, products from others, and certain cloud systems.

Most integrations with Deploy are accomplished by using plug-ins. Plug-ins provide tools for creating component processes and integrations.

See the following page on HCL Plug-ins site for information about plug-ins that support other products: HCL Plug-ins.

Integrating with DevOps Test Performance

When you use HCL DevOps Test Performance (Test Performance), you can create tests and run those tests in HCL DevOps Deploy (Deploy) by using the Test Performance Deploy plug-in.

To learn more about integrating with Test Performance, see documentation.

Administering

Learn how to administer settings in HCL DevOps Deploy (Deploy) elements and communication between the elements.

Backup and recover

You can back up and recover the contents of your HCL DevOps Deploy (Deploy) installation.

Backing up the server

To back up the server, make a backup copy of the database and application data directory. Both frequent online backups and occasional offline backups are required.

About this task

Note the following points before performing the online or offline backup procedure.

Important:

Make sure that you take backups of the filesystem and the database at the same time, or as close as possible. The bigger the gap between backups, the bigger the data skew is likely to be. Using a backup of the server directory and the database taken from different times may cause the following problems when you restore the backups:

- If you back up the file system before the database, then the database may know of the versions or step logs that were created after the file system was backed up. This may lead to problems attempting to deploy those versions or to view those step logs.
- If you back up the database before the file system, then the file system may contain versions and step logs that the database is not aware of. Due to this, you may not get the correct available file system size.

Offline backups

Offline backups happen when the server is stopped, for example during a monthly maintenance window when you install operating system updates and other updates on the server and agent systems.

About this task

Offline backups must include both the database and the server file system, including the application data directory. Follow these steps to take an offline backup of the server:

1. Stop the server.
   See Starting and stopping the server.
2. Create a full backup of the database.
   For instructions on backing up the database, see the documentation for your database.
3. Back up the application data folder to a secure backup location.
4. Back up the `conf` folder.
   On high-availability systems, back up this folder for each server.
5. Start the server.

Online backups

Online backups happen frequently, and without stopping the server. For online backups, you need to back up only the database.

About this task

Online backups, typically include only the database and not the server file system and application data directory. Use the backup options for your database system.

Note: To make the online backups easier to restore, configure your database to create full regular restore points and to enable transaction logging. For example, the database might make full backups once a month and log transactions all the time. That way, you can restore the database at any time.

What to do next

To restore HCL DevOps Deploy (Deploy) from a backup, see Restoring a backup on page 2058.

Restoring a backup

To restore a backup of the server, stop the server and replace the existing files and databases with the backup files and databases.

About this task

You can restore the server, and the app data all at once.

1. Stop the server.
2. Copy the server files from the backup to the application data directory, overwriting the current files.
   You must restore the backup files to the same location as the existing files because configuration files include system paths.
3. Restore the database from the backup.

   Important: You must restore the database from a backup that was taken at the same time as the backup of the application data directory.

4. Restart the server.

High availability and failover

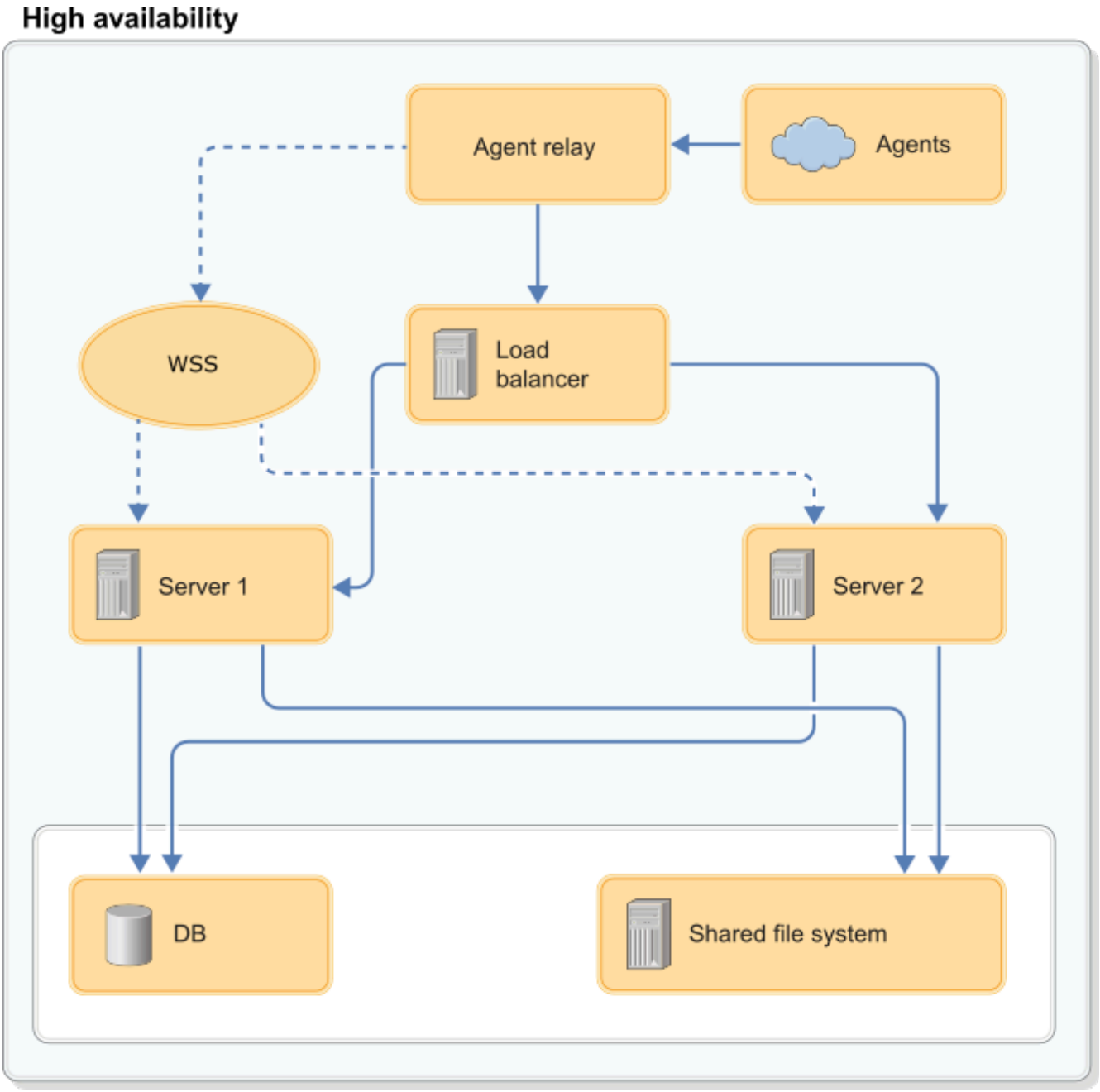
You can configure clustered HCL DevOps Deploy (Deploy) servers, and agents for high availability. You can also configure agents and agent relays for failover.

In most cases, you configure servers to be part of a high-availability cluster when you install them, but you can also convert existing servers to a cluster. To configure a high-availability cluster of Deploy servers, see Setting up high-availability clusters on page 2062.

High availability overview

To configure clustered high availability for HCL DevOps Deploy (Deploy), you must use a load balancer. You can configure a cold standby server by using a load balancer or configuring the servers' DNS.

The architecture of all high-availability scenarios can resemble the following figure:



The load balancer communicates with all Deploy servers. All servers access the same database, shared file system, and agents.

Two common high-availability scenarios are clustered and cold-standby (disaster recovery), and you can apply both scenarios to configurations that use the Deploy server.

Cold Standby

The cold standby option is simple to implement and reduces downtime. In this setup, the load balancer manages two servers, or two servers are connected through DNS settings. Both the primary and secondary servers are connected to

a database and a remote file system. The secondary server is not running. If the active node fails, you must start the secondary server so network traffic can be routed to it. This event is called *failover*. During failover, the secondary server reestablishes connections with all agents, runs recovery, and proceeds with any queued processes. Deploy has no automatic process for failover, but it can be managed by using your load balancer or DNS.

You can convert a cold-standby system to a clustered system by putting shared files on network storage and connecting the servers with cluster connections. See Setting up high-availability clusters on page 2062.

Note: The cold standby server or servers must not be running until they are started by the failover process. Even if no traffic is routed to the cold standby server, it runs internal processes that can corrupt the database. For this reason, servers must be either part of the cluster or shut down.

Clustered

The high availability (HA) feature increases server reliability by distributing processing across a cluster of servers. Each server is an independent node that cooperates in common processing. The goal is to create a fault-tolerant system that requires little or no manual intervention. All servers are running, and the load balancer routes processes to all servers, based on their availability.

All servers recognize each other, and all services are active on each server.

For installation instructions, see Setting up high-availability clusters on page 2062.

High-availability server processes

While high-availability servers complete the same processes as standalone servers, they run slightly differently and access files in run slightly different locations than individual servers do.

In a high-availability topology, all servers use the same database and a common network storage. The server log and CodeStation repository are stored in the common network storage. Most configuration information and runtime data is stored in the shared database. Each server maintains some independent configuration information, such as the connection information for the database.

Because the servers share the database, all servers run on the same interval.

Many activities run the same on high-availability servers as on a standalone server. Some activities run slightly differently:

Importing component versions (CodeStation)

In a clustered high-availability configuration, any server can start the process that imports a component version. If components are configured for automatic imports, all servers poll for component version changes, but they are limited to the polling duration in the system settings. To prevent more than one server from importing a component version or checking for new versions, the database handles server synchronization and a server acquires a database lock before importing or polling.

Notifications

Notifications are handled by the server that runs the process or starts the related event. For more information, see Creating Notifications in a Notification Scheme on page 2081.

Running processes and handling failures

While processes are running, step failures are shown in the process log as usual. Error handling is the responsibility of the process author; for example, processes can include steps that run when other steps fail, including automatic rollback steps.

When a process is requested, a request is created in the database. Servers poll the database regularly for new process requests. The first server that finds the process request runs the requested process. Due to a limitation on the server, the process can not be transferred to another server in the cluster. Therefore, if the server fails in the middle of running a process, the process fails. The other servers in the high-availability cluster run future processes, but they can not take over processes that the failed server was running.

If an agent relay stops while a connected agent is running a process, the agent uses its failover settings to connect to other servers or agent relays. If the agent exhausts its failover options, it continues trying to reconnect. If an agent relay or server comes back online, the agent continues to run the process.

Step failures do not cause agents themselves to fail. If an agent stops while it is running a process, the server assumes that the agent is still running the process and does not mark the process as failed.

High-availability agent communication

Agents communicate with the server over HTTP or HTTPS. These modes of communication are handled differently in a high availability scenario.

HTTP or HTTPS communication

A standard load balancer handles HTTP or HTTPS server-agent communications. Authentication is run on every request; any server can run validation for any request. You configure HTTP or HTTPS server-agent communication when you configure your cluster of servers. See Setting up clusters of servers on page 2063.

In case of server failure, user-server authentication must be redone unless the ‘remember me’ cookie is used.

Setting up high-availability clusters

To configure a high-availability cluster, you store shared files on network storage. Then, you configure multiple servers, set each server to access the same files and database, and configure a load balancer to distribute the traffic between the servers.

About this task

The specific steps depend when you are clustering servers. To set up a cluster of servers, see Setting up clusters of servers on page 2063.

To upgrade a high-availability installation, see Upgrading high-availability installations on page 2229.

Setting up clusters of servers

To configure a cluster of servers, set up a shared database and file system for the servers to use.

Before you begin

- Each server in the cluster must run the same version of Deploy.

About this task

To set up servers in a clustered configuration, you install servers on separate systems and connect the servers to the same database and network storage. Several log and configuration files are stored in the shared network storage, but each server also independently maintains some configuration information, such as database and connection information. The database stores other configuration information and runtime data, as well as other information. Because the servers share the database, all servers run on the same interval.

Note: To create a second server in an high-availability setup use the Deploy installer. An install which copies server install files to create a second server is not supported.

Then, you configure a load balancer to distribute the traffic between the servers. Instead of accessing the servers directly, users access the load balancer URL. To the users, that URL appears to host a single instance of the server with high capacity; users are unaware of the multiple servers.

Note: Servers, agents, and agent relays communicate in several ways: via HTTP/HTTPS on the default ports 8080 and 8443, and via WebSocket connects on the default port 7919. To set up the cluster, you must configure the load balancer to distribute the HTTP/HTTPS traffic. In general, load balancers work well with HTTP/HTTPS traffic because this traffic is stateless and works on a request and response cycle.

1. Set up a shared database for the servers to use.
   See Installing the server database.
2. Set up network storage for the server configuration files.
   Because each cluster server must access the same configuration files, each server must have access to this network storage.
3. Configure online and offline backups for the servers.
   See Backing up the server on page 2057.
4. Install a load balancer to send requests to the servers.
   The load balancer must be able to forward requests to the HTTP and HTTPS ports for the servers. For more information, see the documentation for your load balancer.

   Important: You must configure the load balancer for session persistence, also known as session affinity.

5. If you have one or more servers already installed, populate the network storage with configuration files for the servers:

a. Stop one of the existing servers.

b. On the server, mount the network storage.

c. From the server installation directory, move the following files and folders to the network storage and update the existing server to access those files and folders.

   Which folders you move depends on how the files are structured on the server and which version the server is running on:

   In this case, follow these steps:

   i. Move the application data folder to the network storage.
   ii. On the server, in the `installed.properties` file, update the **server.appdata.dir** property to the new location of the application data folder.

   If the server was upgraded from a version prior to 6.1.1.5 to version 6.1.1.5 or later, the server does not have an application data folder. In this case, you must move the folders into an application data folder on the network storage by running these steps:

   i. On the network storage, create a folder to be the application data folder, such as `/media/share/appdata`.
   ii. Copy the following folders to the application data folder:
      - *`install_folder`*`/var/plugins`
      - *`install_folder`*`/var/repository`
      - *`install_folder`*`/var/sa`
      - *`install_folder`*`/logs`
      - *`install_folder`*`/conf/server/notification-templates`
      - *`install_folder`*`/conf/encryption.keystore`
      - *`install_folder`*`/conf/server.keystore`
      - *`install_folder`*`/conf/collectors`
      - *`install_folder`*`/patches`
      - *`install_folder`*`/conf/server/log4j.properties`

      For *`install_folder`*, use the installation directory for the server.
   iii. Copy (not move) the `<server install>/conf/installed.version` file to the newly created `<appdata>/conf` directory.
   iv. Verify that the application data folder has the correct folder layout. For example, if you use the `/media/share/appdata` folder for the application data folder, make sure that you have the following folders and files:
      - `/media/share/appdata/var/plugins`
      - `/media/share/appdata/var/repository`
      - `/media/share/appdata/var/sa`
      - `/media/share/appdata/logs`
      - `/media/share/appdata/conf/server/notification-templates`
      - `/media/share/appdata/conf/encryption.keystore`

- `/media/share/appdata/conf/server.keystore`
- `/media/share/appdata/conf/collectors`
- `/media/share/appdata/patches`
- `/media/share/appdata/conf/server/log4j.properties`

Some of these files and folders might not exist yet.

v. On the server, in the `installed.properties` file, set the **server.appdata.dir** property to the new location of the application data folder.

d. Delete the original copies of the files and folders that you moved to network storage. (In other words, delete the files on the local server and keep the files on the network storage.)

6. If you terminate SSL at the server level, set the servers to use the same security certificate.

   Example

   For example, you can copy the certificate in the `/opt/tomcat/conf/tomcat.keystore` file to the other servers. Edit the *tomcat.key.alias* property in `installed.properties` file to specify the alias in the keystore that contains the certificate to use.

   ```
   tomcat.key.alias=alias_of_your_certificate_in_tomcat_keystore
   ```

7. **Optional:** Configure the license server for high availability.

   For information, see the Common Licensing documentation. Also, see High availability options for Rational License Key Server.

Results

After you configure the load balancer to distribute connections to the systems, users can connect to a single URL and use the entire cluster. The servers also ensure that the correct number of licenses are used, even if a user accesses multiple servers.

Several server processes behave differently than with individual servers. See High-availability server processes on page 2061.

What to do next

- Add servers to the cluster. See Adding servers to clusters on page 2065.
- Set up cold standby servers to use for disaster recovery. See Adding cold standby servers on page 2068.

Adding servers to clusters

You can add new servers to a cluster at installation time, or you can move existing servers to be part of a cluster.

Installing new cluster servers

The easiest way to add servers to a cluster is to install new servers. The installation program sets them up to be part of the cluster.

Before you begin

- Set up a cluster of servers, including setting up a load balancer, shared database, and network storage. See Setting up clusters of servers on page 2063.
- Each server must run the same version of HCL DevOps Deploy (Deploy).

1. On the system on which you are installing the new server, set up a connection to the network storage.
   Example
   For example, on Linux™ systems, you might mount a remote drive to the local file system.
2. Install the server as described in Installing the server in interactive mode with the following details:
   - When the installation program asks for database information, specify the shared database for the cluster.
   - When the installation program asks for the host name for users to access, specify the host name of the load balancer.
   - When the installation program asks whether the server is part of a high-availability cluster, specify `yes`.
   - When the installation program asks for the location of the application data, specify the same folder on the network storage that the other cluster servers are using. If this is the first server in the cluster, you can specify any folder on the network storage. In this case, other servers that you add to the cluster must also use this folder.
3. Set the new server to use the same security certificate as the existing servers.
   Example
   For example, you can copy the certificate in the `/opt/tomcat/conf/tomcat.keystore` file to the other servers. Edit the *tomcat.key.alias* property in `installed.properties` file to specify the alias in the keystore that contains the certificate to use.
   ```
   tomcat.key.alias=alias_of_your_certificate_in_tomcat_keystore
   ```
4. Configure the load balancer to send HTTP/HTTPS traffic to the new cluster server.

Moving existing servers to a cluster

Before you begin

- Set up a cluster of servers, including setting up a load balancer, shared database, and network storage. See Setting up clusters of servers on page 2063.
- Each server must run the same version of HCL DevOps Deploy (Deploy).

1. On the system that hosts the server, set up a connection to the network storage.
   Example
   For example, on Linux™ systems, you might mount a remote drive to the local file system.
2. If the sever contains elements that you want to keep, such as components, applications, or processes, export those elements and import them into an existing cluster server.
3. Stop the server.
4. Edit the `installed.properties` file on the server, located in the *`install_folder`*`/conf/server` directory.

a. Set the **server.appdata.dir** property to the location of the application data folder that the cluster servers are using.

b. Copy the value of the **encryption.keystore.alias** property to the equivalent file on the new cluster server from a server that is already in the cluster.

c. Add the following line of code to the file:

```
com.urbancode.ds.UDeployServer.multiserver=true
```

d. Update the database settings to use the same database as the other servers in the cluster.

5. Set the new server to use the same security certificate as the existing servers.
   Example
   For example, you can copy the certificate in the `/opt/tomcat/conf/tomcat.keystore` file to the other servers. Edit the *tomcat.key.alias* property in `installed.properties` file to specify the alias in the keystore that contains the certificate to use.

```
tomcat.key.alias=alias_of_your_certificate_in_tomcat_keystore
```

6. Start the server.
7. On the new cluster server, click **Settings > System Settings**, and set the **External Agent URL** and the **External User URL** to the URL for the load balancer.
8. Click **Save**.

Converting a stand-alone server to high availability

You can convert an HCL DevOps Deploy (Deploy) server into the first node of a high-availability cluster.

Before you begin

- Each server in the cluster must run the same version of Deploy.

About this task

To convert a stand-alone server to the first node in a clustered configuration, you first move CodeStation to shared network storage, and then modify the server settings for multi-server operation.

1. Move CodeStation to a shared storage location with a fast network connection.
   Because each cluster server must access the same configuration files, each server must have access to this network storage. See Relocating CodeStation on page 2124.
2. Install a load balancer to send requests to the servers.
   The load balancer must be able to forward requests to the HTTP and HTTPS ports for the servers. For more information, see the documentation for your load balancer.

Important: You must configure the load balancer for session persistence, also known as session affinity.

3. Stop of the server.
4. Add the following line of code to the *install_folder*/conf/server/installed.properties file:

```
com.urbancode.ds.UDeployServer.multiserver=true
```

5. Start the server.
6. On the new cluster server, click **Settings > System Settings**, and set the **External Agent URL** and the **External User URL** to the URL for the load balancer.

Results

After you configure the server and the load balancer, you can add nodes to the cluster.

What to do next

- Add new servers to the cluster. See Adding servers to clusters on page 2065.

Adding cold standby servers

You can add cold standby servers to serve as a backup server or for disaster recovery. You must start the secondary server or servers when the active server fails, and network traffic is routed to the secondary servers.

Before you begin

- Make sure that the server is connected to a license server with available licenses.
- Set up a cluster as described in Setting up clusters of servers on page 2063. Your cluster must contain at least one running server.

About this task

To set up a cold standby system, you install two or more servers on separate systems and connect the servers to the same database and network storage. Then, you configure a method to distribute the traffic to secondary servers if the primary servers fail. In most cases, you must manually start the secondary servers after the primary servers fail.

During failover, the secondary server reestablishes connections with all agents, runs recovery, and proceeds with any queued processes. Deploy has no automatic process for failover, but it can be managed by using your load balancer or DNS.

You can use cold standby servers with or without a load balancer. If you use a load balancer, instead of accessing the servers directly, users access the load balancer URL. To the users, that URL appears to host a single instance of the server with high capacity; the users are not aware of the multiple servers.

1. Install the cold standby server:
   Install the server as described in Installing the server in interactive mode with the following details:

- When the installation program asks for database information, specify the database that the cluster servers are using.
- If you are using a load balancer, when the installation program asks for the host name that users will access, specify the host name of the load balancer.
- When the installation program asks if the server is part of a high-availability cluster, say yes.
- When the installation program asks for the location of the application data, specify the same folder on the network storage that the other cluster servers are using.

Note: Do not start the server.

2. Set the new server to use the same security certificate as the existing servers.
   Example
   For example, you can copy the certificate in the `/opt/tomcat/conf/tomcat.keystore` file to the other servers. Edit the *tomcat.key.alias* property in `installed.properties` file to specify the alias in the keystore that contains the certificate to use.

   ```
   tomcat.key.alias=alias_of_your_certificate_in_tomcat_keystore
   ```

3. Set a method to route traffic to the cold standby server after failover.
   Depending on your network configuration, you might configure a DNS server routing path, Virtual IP, cluster connection, or load balancer to swap traffic.

Server settings and configuration

After you install HCL DevOps Deploy (Deploy), you can modify the server configuration and configure more options.

Other server settings control locks, notifications, and general system settings.

Increasing the amount of memory that is available to the server

If you see "Out of memory" errors or are using the server at high capacity, increase the amount of system memory that is available to the server.

About this task

For servers that are running as a service on Windows™, see the following technote: Out of memory errors while running IBM UrbanCode Deploy.

To increase the memory for the server on Windows™ for a non-service installation, or on Linux™ or UNIX™ systems, follow these steps:

1. Stop the server.
2. In the `bin/set_env.cmd` file (Windows) or `bin/set_env` file (Linux and UNIX), increase the memory in the `JAVA_OPTS` parameter.
   Example
   For example, the current file might have a line that begins with the following code:

```
JAVA_OPTS="-Xmx2048m
```

To increase the memory, change the parameter that begins with the code `-Xmx`. For example, to increase the memory to 4 GB, use the following value: `-Xmx4096m`. Be sure to include the "`m`" at the end of the parameter.

3. Save the file and restart the server.

Modifying the context root of the server

After you install the server, you can modify the context root, if your server configuration requires it.

1. Stop the server.
2. From the server installation directory, rename the directory *`server_install`*`/opt/tomcat/webapps/ROOT` to a new name, such as *`server_install`*`/opt/tomcat/webapps/deploy`
   Use the server installation directory for *`server_install`*.
3. Make a backup copy of the file `opt/tomcat/conf/server.xml`.
4. Open the file `opt/tomcat/conf/server.xml` in a text editor
5. In the `server.xml` file, find the following line of code:

```
<Context path=""
  docBase="ROOT"
  debug="0"
  reloadable="false"
  useHttpOnly="true"
  sessionCookieName="JSESSIONID_8080">
```

6. In this code, change both the **docBase** attribute and the **path** attribute to the name of the new folder.

   Example

   For example, if you renamed the `ROOT` folder to `deploy`, the code looks like the following example:

```
<Context path="deploy"
  docBase="deploy"
  debug="0"
  reloadable="false"
  useHttpOnly="true"
  sessionCookieName="JSESSIONID_8080">
```

7. Save the file.
8. Restart the server.

Results

Now you can access the server at the following URL:

```
https://hostname:port/foldername
```

Use the new name of the folder as *`foldername`*.

Management of maintenance blackouts

You can manage important updates to ensure that tasks run smoothly without interfering with ongoing deployments by using maintenance blackouts. You can manage maintenance blackouts by creating a recurrence blackout, rescheduling an existing blackout, creating a specific environment blackout, and canceling an existing blackout.

You can schedule a maintenance blackout in the daily, weekly, or monthly recurrence pattern. When a recurring scheduled deployment is created, its events are now linked together into a series.

You can update a maintenance blackout by modifying the name and description and rescheduling an upcoming maintenance blackout of a recurrent deployment series.

Rescheduling an upcoming recurring maintenance blackout changes the timing of subsequent events in the same series. For example, if the daily blackout series is moved from 7 pm to 8 pm, all future blackouts are also moved to 8 pm. Similarly, if the blackout is extended from 1 hour to 2 hours, the duration of all subsequent blackouts is extended to 2 hours.

Note: Suppose a maintenance blackout is rescheduled to a later date, any events that were scheduled before that date are removed. Recurrences that were canceled on a specific date remain canceled. For example, if the blackout event for next Monday is canceled due to a holiday, and later the Tuesday blackout is rescheduled to Wednesday, then the blackout on Monday remains canceled.

If a recurring deployment is rescheduled, the dates and cancellations of future deployments are adjusted in the same way as recurring maintenance blackouts. However, if the recurrence pattern is changed, then the cancellations are reset.

Entire recurrence series or individual events can be canceled without affecting the rest of the events in the series.

Related information

Creating recurrence maintenance blackouts on page 2071

Updating maintenance blackouts on page 2072

Canceling maintenance blackouts on page 2074

Creating recurrence maintenance blackouts

You can schedule a maintenance in a daily, hourly, weekly, or monthly recurrence pattern.

Before you begin

You must ensure the following requirements are met:

- Obtained permission to manage maintenance mode.
- Enabled the maintenance mode.

About this task

You cannot alter the recurrence pattern of maintenance blackouts after you created a pattern.

1. Log in to Deploy.
2. Go to **Settings**.
3. Click **Manage Maintenance Blackouts** in the **System** panel.
   The settings **System** page is displayed.
4. Click **Create Maintenance Blackout**.
   The **Create Maintenance Blackout** window is displayed.

5. Enter the name or description in the **Name/Description** field.
6. Select the **Start Date** for the blackout.
7. Select the **Start Time** from the drop-down.
8. Select the **Estimated End Date** from the drop-down.
9. Select the **Recurrence Pattern** from the drop-down.
10. You can choose the **Hourly**, **Daily**, **Weekly**, **Monthly**, and **Weekdays** recurrence pattern.

    Note: Select **Never** for a one-time blackout event.

11. Click **Submit**.

Results

You have created a recurrence maintenance blackout. The new recurrence blackout is displayed in the maintenance blackout list.

What to do next

You can view the recurring maintenance blackout in on the **Calendar** page.

Related information

Enabling maintenance mode on page 2073

Management of maintenance blackouts on page 2070

Updating maintenance blackouts

You can change the name and description of the past and upcoming recurring blackouts in a series.

Before you begin

You must ensure the following requirements are met:

- Obtained permission to manage maintenance mode.
- Enabled the maintenance mode.

About this task

You can update a maintenance blackout by modifying the name and description and rescheduling an upcoming maintenance blackout of a recurrent deployment series.

Note: You cannot change the name and description of deployments beyond the upcoming maintenance blackout.

1. Log in to Deploy.
2. Go to **Settings**.
3. Click **Manage Maintenance Blackouts** in the **System** panel.

The settings **System** page is displayed.

4. Click and then click **Edit** for the maintenance blackout you want to reschedule.
   The **Update Maintenance Blackout** window is displayed.
5. To modify the name and description, enter a name and description in the **Name/description** field.
6. To reschedule an upcoming maintenance event, select the **Start Date** for the blackout.
7. Select the **Start Time** from the drop-down.
8. Select the **Estimated End Date** from the drop-down.
9. Select the **Recurrence Pattern** from the drop-down.
   You can choose the **Daily**, **Weekly**, or **Monthly** recurrence pattern.

   Note: Select **Never** for a one-time blackout event.

10. Click **Submit**.

Results

You have updated a recurrence maintenance blackout by modifying the name and description and rescheduling the event. The updated maintenance blackout is displayed in the maintenance blackout list.

What to do next

You can view the updated maintenance blackout in on the **Calendar** page.

Related information

Enabling maintenance mode on page 2073

Management of maintenance blackouts on page 2070

Enabling maintenance mode

Enabling the maintenance mode stops some activities on the server for a specific duration so you can run important tasks that require the server to be idle.

Before you begin

Ensure that you have the permissions to manage maintenance mode.

About this task

When you enable maintenance mode, users can log in to the server as usual and perform some limited tasks. A message is displayed at the top indicating the start and estimated end time of the maintenance activity. While the maintenance mode is enabled, some of the tasks/actions are blocked and some are allowed. Below is a list of such tasks:

Tasks blocked during maintenance mode	Tasks not blocked during maintenance mode
Execution of new global, component, and application processes.	Processes which are started before maintenance mode are allowed to run for completion.

Tasks blocked during maintenance mode	Tasks not blocked during maintenance mode
With additional permissions, you can allow users to execute new deployments while maintenance mode is enabled.	
Execution of processes triggered by the calendar, including recurring deployments.	Sub-processes kicked off by processes that are started before maintenance mode are allowed.
Execution of processes triggered by a repeating a process request.	Manual tasks that are part of a process can be responded to, and the process are allowed to continue.
Approval or rejection of any approval tasks.	Making changes to the server, such as changing properties, creating elements such as applications and resources, and editing processes.
New version imports.	-
Automatic version imports.	-
Processes started upon completion of a version import.	-
Auto Discovery.	-
Auto Compare.	-
Auto Configure.	-

1. Click **Settings > System Settings**.
2. Activate the **Enable Maintenance Mode** option in the **General Settings** field.
3. **Save** the settings.

Results

The maintenance mode is enabled.

What to do next

- After maintenance mode is enabled, make sure that you do not restart the server until all processes have completed.
- After the maintenance schedule ends, you must disable the **Enable Maintenance Mode** option manually.

Canceling maintenance blackouts

You can cancel the entire recurrent deployment series or just a single deployment blackout event without impacting the other deployments in the recurrence series.

Before you begin

You must ensure the following requirements are met:

- Obtained permission to manage maintenance mode.
- Enabled the maintenance mode.

About this task

Note: You cannot revive the cancelation of the entire recurrent deployment blackout series or a single deployment blackout event.

1. Log in to Deploy.
2. Go to **Settings**.
3. Click **Manage Maintenance Blackouts** in the **System** panel.
   The settings **System** page is displayed.
4. Click and then click **Cancel** for the maintenance blackout you want to cancel.
   The **Cancel Series or Event** window is displayed.
5. Select the **Event** or **Series** you want to cancel.
6. Click **Submit**.

Results

You have canceled the maintenance blackout event or series.

Related information

Enabling maintenance mode on page 2073

Management of maintenance blackouts on page 2070

Changing the database password

If you change the password for your database, you must update the password on the server.

1. In a text editor, open the file `conf\server\secured-installed.properties`.
2. Specify the new database password in the following line of code:

   ```
   hibernate.connection.password=newPassword
   ```

   This password will be encrypted when you restart the server.
3. Restart the server.

Changing passwords for the server KeyStore

You can change the Java™ KeyStore password for the server.

About this task

The server Java™ KeyStore password must be identical to its certificate password. After you change both the server KeyStore password and certificate password, you must update the password in the server properties.

1. Stop the server.
2. Open a command-line window, and go to the *appdata*`/conf` directory.

3. Change the server KeyStore password by using this command:

   ```
   keytool -storepasswd -new newStorePassword -keystore server.keystore
   -storepass changeit
   ```

   The default server password is *changeit*. The keytool application is included in the Java™ developer kit and is not part of HCL DevOps Deploy (Deploy).
4. Change the certificate password by using this command:

   ```
   keytool -keypasswd -alias server -keypass changeit -new newKeyPassword -keystore server.keystore
    -storepass newStorePassword
   ```

   The default server alias is *server*.
5. In a text editor, open the *server-install*\conf\server\secured-installed.properties file.
6. Specify the new Java™ KeyStore password for the server in the following line of code:

   ```
   server.keystore.password=newStorePassword
   ```

   The `javax.net.ssl.trustStorePassword` property takes precedence over the `secured-server.keystore.password` property. If it is present in *server-install*\conf\server\secured-installed.properties file, update its password with this code: `javax.net.ssl.trustStorePassword=newpassword`. This property applies to all relevant KeyStores.
7. Specify the new Certificate password for the server in the following line of code:

   ```
   server.key.password=newKeyPassword
   ```

8. Start the server.

Results

The server KeyStore password is changed and is obfuscated in the *server-install*/conf/server/secured-installed.properties file.

Changing passwords for the encryption KeyStore

You can change passwords for the encryption KeyStore.

About this task

The `encryption.keystore` password must be identical to the certificate password.

1. Stop the server.
2. Open a command line window, and go to the *app_data*/conf directory.
3. To view the alias of the secret key, run the following command:

   ```
   keytool -keystore encryption.keystore -list -storetype pkcs12 -storepass changeit
   ```

   Make a note of the alias of secret key, it is required in step 6 on page 2077. The keytool application is included in the Java™ developer kit and is not part of HCL DevOps Deploy (Deploy).
4. Back up the `encryption.keystore` file.
5. Change the encryption KeyStore password by using the following command:

   ```
   keytool -storepasswd -new newPassword -keystore encryption.keystore -storetype pkcs12 -storepass changeit
   ```

The default password is **changeit**. Replace *newPassword* with your password of choice. The keytool application is included in the Java™ developer kit and is not part of Deploy.

6. Change the secret key password by using the following command:

```
keytool -keypasswd -alias XXX -keypass changeit -new newpassword -keystore encryption.keystore
-storetype pkcs12 -storepass newpassword
```

where `alias` is the one that you noted in step 3 on page 2076, and the encryption KeyStore password and the secret key password must be the same.

7. In a text editor, open the *server-install*`\conf\server\secured-installed.properties` file. Specify the new **encryption.keystore** password in the following line of code:

```
encryption.keystore.password=newPassword
```

Note: The password that you specify here must be the same as the password for the encryption KeyStore password that you specified in step 5 on page 2076 and the secret key password that you specified in step 6 on page 2077.

8. Start the server.

Results

Starting the server automatically encrypts the `encryption.keystore` password that you specified in the `installed.properties` file.

Changing passwords for the Tomcat keystore

You must change the default passwords for the Tomcat keystore and private keys.

About this task

The default password for Tomcat private key and keystore is *changeit*. You must change the passwords by using the `keytool` command.

Note: The keytool application is included in the Java™ developer kit and is not part of HCL DevOps Deploy (Deploy).

If you change the key password to a value other than the keystore password, you must update the password in the `secured-installed.properties` file manually. The properties for Tomcat key and keystore passwords are defined as **tomcat.key.password** and **tomcat.keystore.password** in the `secured-installed.properties` file.

The properties are referenced in the `server/opt/tomcat/conf/server.xml` file as `keyPass="${tomcat.key.password}"` and `storePass="${tomcat.keystore.password}"` in the HTTPS connector configuration.

Important: If you use a plain text password in the `server.xml` file and the passwords entered in the `secured-installed.properties` and `server.xml` files do not match, then the server fails to accept any web requests

and you cannot access the application on your browser. As a Tomcat keystore limitation, you do not get any visible error messages.

You must consider the following scenarios before changing the passwords:

- If you change the keystore password, then you must change the Tomcat private key password as well.
- If the keystore has a typical configuration where no explicit key password is set, then you can just change the keystore password to change them both.
- You can directly change the key password only in advanced scenarios where the key password is already different from the store password.
- You are responsible for managing your passwords. Deploy currently does not include the password recovery feature.

When you are upgrading from version 8.1.0 or before, the plain text passwords in the `server.xml` file are replaced with the `${tomcat.key.password}` and `${tomcat.keystore.password}` variables, and the passwords are added to the `secured-installed.properties` file automatically.

1. Stop the server.
2. Open a command-line window, and go to the `opt/tomcat/conf` directory.
3. Change the Tomcat keystore password by using the following command:

   ```
   keytool -storepasswd -new newStorePassword -keystore tomcat.keystore -storepass changeit
   ```

4. Change the Tomcat private key password by using the following command:

   ```
   keytool -keypasswd -alias server -keystore tomcat.keystore -storepass changeit -keypass changeit
   ```

   Notes:
   - The default server alias is *server*.
   - In this example, the password for keystore is assumed to be the default password.
   - The passwords for the `keypass` and `storepass` attributes must be correct.

5. You are prompted to enter a new password:

   ```
   New key password for <server>:
   Re-enter new key password for <server>:
   ```

   A warning message is displayed:

   ```
   The Java Keystore (JKS) keystore uses a proprietary format. If you are using
   JKS keystore, you must migrate to Personal Information Exchange Syntax
   Standard (PKCS12) which is an industry standard format using the following
   keytool command:
   keytool -importkeystore -srckeystore opt/tomcat/conf/tomcat.keystore
   -destkeystore opt/tomcat/conf/tomcat.keystore -deststoretype pkcs12
   ```

6. In a text editor, open the *`server-install`*`\conf\server\secured-installed.properties` file.
7. Specify the new key and keystore password for the Tomcat in the following lines of code:

```
tomcat.key.password=newKeyPassword
tomcat.keystore.password=newKeystorePassword
```

8. Start the server.

Results

The Tomcat key and keystore passwords are changed and are obfuscated in the *server-install*/conf/server/secured-installed.properties file.

Changing or updating the JRE of servers

You can change the JRE of active servers by editing their configuration files.

About this task

Procedure

Note:

- Any certificates that you imported to the old JRE cacerts file must be imported into your new JRE cacerts file as well. SeeConfiguring SSL on Apache Tomcat and LDAP servers on page 2159 for more information.
- Starting from v7.2.1, the keystore converter utility is no more needed, and Deploy handles any necessary conversions itself.

1. Install the new JRE.
2. Stop the server.
3. Update the location of the JRE in the following files:
   - In the file *server_install*/bin/set_env, set the value of the **JAVA_HOME** system variable to the location of the JRE.
   - In the file *server_install*/conf/server/installed.properties, set the value of the **install.java.home** property to the location of the JRE.

Note: As a shortcut, you can delete the old JRE and move the new JRE into the location of the old JRE. In this case, you do not need to update the **JAVA_HOME** system variable and **install.java.home** property because they now point to the new JRE.

Note: The value of the **JAVA_HOME** system variable must match the value of the **install.java.home** property.

Note: If you are upgrading to Java 9 or above and are moving off Oracle Java to AdoptOpenJDK you must manually remove the `-Djava.endorsed.dir` entry from the server_install/bin/set_env file.

4. If the server is running as a service on Windows:™

a. Open the file *server_install*\bin\service_service.cmd in a text editor.

b. Edit the line that starts with the code `set JAVA_HOME` to the path to the new JRE.

c. Open a command prompt as an administrator.

d. Go to the *server_install*\bin\service folder.

e. Remove the service by running the following command:

```
service.cmd remove service_name
```

Use the name of the service for *service_name*. To find the service name, open the services view and find the entry that starts with `Deploy`. The service name is in parentheses in this entry.

f. Reinstall the service by running the following command:

```
service.cmd install service_name
```

5. Start the server.

Working with Notifications

HCL DevOps Deploy (Deploy) can send email notifications whenever user-defined trigger events occur on the server.

Notifications can be sent to different users during deployment, as a way of providing updates to the status of the deployment. For example, notifications can be sent when a deployment finishes or if an approval was accepted or rejected.

Notifications are defined in Notification Schemes. Notification Schemes allow multiple notifications to be configured and assigned to a process. Refer to Creating Notifications in a Notification Scheme on page 2081.

Notification-Approval Relationship

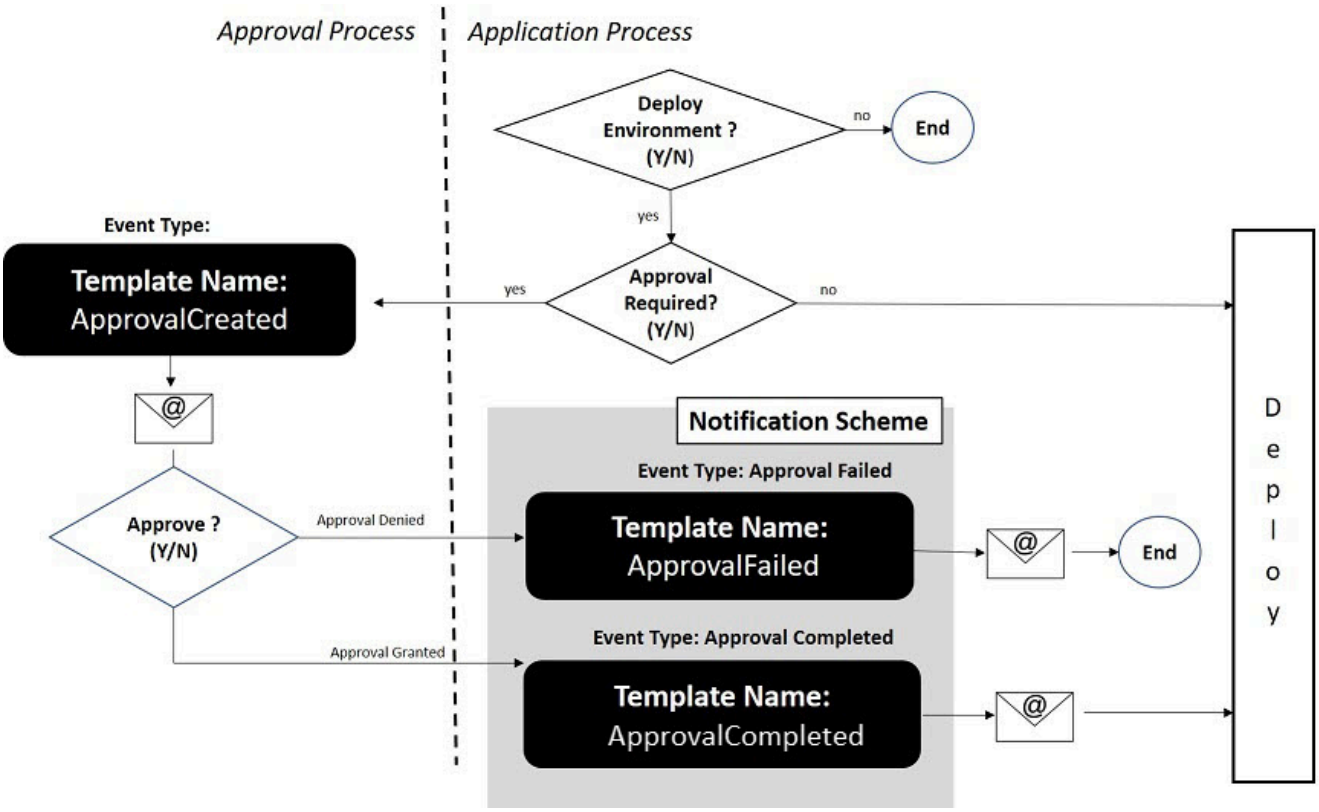
Notifications and approvals are two separate areas of functionality and are configured in two separate areas in the user interface. They are also typically configured by different types of users.

Note: Do not confuse creating the notification for an approval with actually creating the approval itself. Refer to Creating an approval process on page 2388.

Notifications are managed at a system-wide level, typically by the system administrator and can be shared across applications. Notification emails are provided for informational purposes only, and do not affect the execution of the deployment. They do not require user intervention.

Approvals are managed at an application level and require an environment developer user role. Approvals, unlike notifications, are actionable events. They are configured in an approval process and require user-intervention (acceptance or rejection of a deployment in a specific environment). If approved, the deployment executes. If rejected, the deployment remains in an on-hold status.

The only relationship between notifications and approvals is when an action defined in an approval process triggers an approval notification email.



Creating Notifications in a Notification Scheme

Notifications are defined in a notification scheme. The notification scheme is assigned to an application and the notifications are sent during the deployment of that application.

Before you begin

Each notification in a notification schemes must specify the group of users that receive the notifications. Each user in the group must have a valid email address defined on the HCL DevOps Deploy (Deploy) server. An external SMTP mail server is also required.

Prior to configuring notifications, the following prerequisites must be met.

Table 679. Notification Prerequisites

Prerequisite	Description
An external SMTP mail server is required to send notifications.	For information about configuring a mail server, see System settings on page 2086

Table 679. Notification Prerequisites (continued)

Prerequisite	Description
Notification recipients must have defined security system roles.	See Managing Security on page 2149
Notification recipients must have email addresses defined on the Deploy server.	See System settings on page 2086 Note that email addresses are not automatically imported from authentication servers such as LDAP servers. Click **Settings > Authentication**, edit a user, and ensure that the user has an e-mail address attached.

About this task

Notification schemes can be configured to hold multiple notifications related to an application. Since notification schemes are a system-resource, they and can be shared across applications for ease of use.

Each notification you configure in a notification scheme must specify the following items: a notification template, a triggering event, and a user group role to receive the notification. The user group role is inherited from the security system. For example, it is common for an administrator or environment owner to be notified when a work item is generated.

Note: Notifications are **not** sent to users assigned to any role on the System Team, regardless of whether or not that role is configured to receive notifications for the application.

1. Create the new notification scheme. Later, you can come back to add more notifications to the scheme or edit existing notifications. Display the **Notifications** pane. Click **Settings > Notification Schemes**
2. Click **Create Notification Scheme** and provide a **Name** and **Description** for the new notification scheme. (To edit an existing notification scheme, select that scheme in the list.)
3. Configure a notification for the notification scheme. Click **Add Notification Entry**, and supply parameters for the notification at the **Add Notification Entry** window. Notification parameters are listed in the following table.

Table 680. Notification Parameters

Parameter	Description	Values
Role	**Role** corresponds to the roles in the security system. The notification message goes to any user who has the specified role on a team that is associated with the **Target** element. For example, a deployment engineer would be interested in being notified about a failed deployment.	*<user group role>*
Event Type	This field lists the type of event that will trigger the notification.	Select these event types for generic and application processes: **Process Success**, **Process Failure**, or

Parameter	Description	Values
	For example, you can trigger a notification when a processes completes successfully. You can also trigger a notification when an approval is declined. Note that the **Event Type** selected should correspond to the type of process (generic, application, environment, resource, or approval) that triggers the event. Refer to Processes on page 2398 and Creating an approval process on page 2388.	**Process Started**. Select these event types for approval processes: **Approval Completed** or **Approval Failed**.
Target	The **Target** specifies the area of the deployment where the **Event Type** occurs. The target refers to the general concept of an application, environment, or resource in the deployment.	**Application**, **Environment**, or **Resource**
Type	The target can be further refined using the **Type** field. The **Type** pull-down lists the security types defined in a role's application, environment or resource permissions. Notifications will only be sent to users with roles that are configured with the selected security type indicated in the **Type** pull-down. Refer to Creating roles and assigning permissions on page 2188. For example, if the **Target** field is set to **Application**, notifications are only sent to users that are configure in that specific application permission security type when the application is deployed.	*<security type>*

Parameter	Description	Values
Template Name	Default notification templates are provided for use. The templates contain fields that are automatically populated and formatted in the email when the notification is sent to the selected role. The template you select should be based on the **Event Type**, or reason for the notification. You can select a default notification template or a notification template you have created. See Creating notification templates on page 2587. Note that the **ApprovalCreated** notification template is NOT intended for use in a notification scheme. It appears in the **Template Name** drop-down, but should only be assigned as a single template when Creating an approval process on page 2388 to notify an approver that an approval is required.	Use these default notification templates for **Process Success**, **Process Failure**, or **Process Started** event types. ◦ **ProcessRequestStarted** - indicates that a specific application, user group environment, or application resource was initiated. ◦ **ApplicationDeploymentSuccess** or **ApplicationDeploymentFailure**- indicates that a specific application deployment succeeded or failed. ◦ **TaskCreated** - reports on the state of manual tasks. ◦ **DeploymentReadied** - notifies the role that the deployment requires approval. ◦ **GenericProcessRequestStarted** - indicates that a Generic processes on page 2451 was started. ◦ **GenericProcessSuccess** or **GenericProcessFailure** - indicates that a Generic processes on page 2451 succeeded or failed. Use these notification templates for **Approval Completed** or **Approval Failed** event types. ◦ **Deployment Readied** - indicates that an approval was granted for an approval process. ◦ **ApprovalFailed** - indicates that approval was denied for an approval process.

4. Assign a notification scheme to an application or approval process. For an application process, click the application, go to the **Configuration** tab, and in the **Notification Scheme** list, select the notification scheme. For an approval process, click the application, go to the **Approval Process** tab, edit the required process step and assign the approval notification template at the **Edit Properties for Component Approval Task** window.

Managing locks

A *lock* ensures that processes do not interfere with one another. Normally, locks are released after they are no longer needed. Sometimes, a lock is not released, and its associated process is unable to complete. The lock management feature helps you to identify and resolve abnormal lock conditions.

About this task

Follow these steps to view and resolve locks.

1. A running process with a lock, like all active processes, is shown on the **Dashboard** tab with a status of `Running`. If a locked process takes longer to complete than expected, you can cancel the process from the **Dashboard**, or investigate it fully with the **Settings** tab.
2. Display the **Locks** pane by clicking the **Locks** link on the **Settings** tab (**Settings > Locks**).
   Result
   The **Locks** pane displays the following information:

   Table 681. Lock fields

Field	Description
Name	The name identifies the lock. The displayed name is a concatenation of the component or application name (depending on type) + process name + resource name.
Type	Indicates whether the process that created the lock is a component or application type. Locks can be applied only to component or application processes.
Component/Application	Displays the name of the component or application that contains the lock. Clicking an item displays (depending on the type) the **Component** pane, or **Application** pane, where you can investigate the lock.
Resource/Environment	Displays the name of the resource or environment that contains the lock. Clicking an item displays (depending on the type) the **Resource** pane, or **Environment** pane.
Process	Displays the name of the process that contains the lock. Clicking an item displays the process in the process editor.
Actions	Lists the available actions.

3. Resolve the lock by selecting an action:

   Table 682. Lock actions

Action	Description
View Request	Displays the process log for the process that contains the lock. You can use the **Actions** field on the displayed pane to see the name of the process step that is causing the lock.
Release	Releases the lock, which enables the associated process to continue processing.

If the HCL DevOps Deploy (Deploy) server and or agents go down while a locked process is running, Deploy automatically restores any interrupted processes along with any locks they might contain after service is restored.

Acquire Lock on page 2466

Release Lock on page 2483

System settings

System settings allows you to manage global settings for components, clean up, password requirement, security, mail-server, and much more.

Example

The following tables list the settings for several system-wide settings for the server.

Table 683. General settings

Field	Description
External Agent URL*	Specify the URL that the HCL DevOps Deploy (Deploy) server provides to agents as their point of contact. When an agent wants to contact the server, it retrieves this URL and opens a connection on it. This URL must point to the HTTP or HTTPS port of the server.
External User URL*	Specify the URL that users enter to connect to the Deploy server. The URL is also used when creating a notification template.
Message of the Day	A custom notification message that is displayed when users log in to the dashboard. This field supports standard Markdown syntax, which you can use to include formatted headings, bold text, numbered or bulleted lists, inline code blocks, and internal or external hyperlinks.
Validate Agent IP	When enabled, the agent IP and host name are validated against values that are used the first time that the agent is connected to the Deploy server.
Create Default Children for Plugins	When enabled, empty groups are created when plug-ins use the auto discovery feature.
Default Locale	Select a default language for the user interface.
Default Snapshot Lock Type	Determines the default lock type when users create snapshots. Lock types are the following values: **Only Component Versions**, **Only Configuration**, or **Component Versions and Configuration**. The default value is **Component Versions and Configuration**. Changes are effective after the server is restarted.
Require a Comment For Process Design Changes	When enabled, prompts users to provide a comment when a process design is updated.
Enable Maintenance Mode	When enabled. stops activities on the server so you can run important tasks that require the server to be idle.

Table 684. Component settings

Field	Description
Components copy to CodeStation by default	When enabled, newly created components copy artifacts to CodeStation by default. You can change this setting on individual components even when this option is enabled.
Enforce Deployed Version Integrity	When enabled, deployed component version artifacts cannot be changed.
Automatic Version Import Check Period (seconds)*	Specify the time-period (in seconds) between the times that Deploy polls components for new versions. If this value is changed, the server must be restarted before the change becomes effective. You can set this value to 15 seconds or higher.
Use Agent Tag For Integration	When enabled, allows agent that is identified by a specific tag to import component versions. If this setting is selected, the **Agent Tag for version imports** field is shown. Otherwise, the **Agent for Version Imports** field is displayed. For information about tagging objects, see Adding tags to objects on page 2521
Agent for Version Imports	Specify the agent that is used to import component versions. The agent must have access to component artifacts. The agent becomes the default agent for new components. This value can be overridden when a component is created. This field is displayed if the **Use Agent Tag for Integration** toggle button is set to Off. To import artifacts from the server location, install an agent in the same location and specify that agent.
Version Import Records to Retain	Specify the number of version import records to retain during cleanup. Default is 100.
Enable Safe Edit of Component Processes	When enabled, you can test component processes before making them available in normal environments. You can make updates to your component process, test those updates safely, and then promote to a new version when you are ready.
Require Approval to Promote	When enabled, any draft component process must be approved by an authorized user before it can be promoted and used in deployments.

Table 684. Component settings (continued)

Field	Description
Component Process	
Allow Process Locking	When enabled, users with 'Manage Process Lock' permission can lock component processes for editing. Users can also force unlock a process locked by another user.
Require Process Locking	When enabled, users will not be able to edit component processes without locking the process first.
Server Storage Display '100%' Size (MB)	Define the size in megabytes of '100%' storage in the dashboard server storage display. This setting only affects the dashboard display and does not enforce a limit for artifact uploads.

Table 685. Artifact cleanup settings

Field	Description
Daily Cleanup Start Time*	Enter the time when versions are cleaned. The value must be an integer from 0 (midnight) to 23 (11 pm). To preview the component versions that are scheduled for cleaning, click **Preview Version Cleanup**. See Cleaning up component versions on page 2136.
Default Application Snapshot Retention (days)*	Enter the number of days application snapshots are kept by default. A value of `-1` means that they are kept indefinitely. Component and environment settings can override this setting.
Default Version Retention for Component (days)*	Enter the number of component versions to keep by default. A value of `-1` specifies that all are kept. Component settings can override this setting.
Default Number of Versions to Retain for Component*	Enter the number of versions to retain for each component.
Default Version Retention for Environment (days)*	Enter the number of environment versions to keep by default. A value of `-1` specifies that all are kept. Environment settings can override this setting.
Default Number of Versions to Retain for Environment*	Enter the number of versions to retain for each environment.

Table 685. Artifact cleanup settings (continued)

Field	Description
Archive Path	Specify the path where the compressed file that contains the archived component versions is written. If this field is blank, the compressed file is not written, and no archive is kept.
Preview Version Cleanup	Click **Preview Version Cleanup** to see the versions that are archived using the current settings. See Cleaning up component versions on page 2136.

Table 686. Process history cleanup settings

Field	Description
Daily Cleanup Start Time	Specify the time that the daily cleanup process starts.
Daily Cleanup Duration (hours)	Specify the number of hours that the cleanup process runs daily.
Enable Deployment History Cleanup	When enabled, deployment history files are reduced. See Cleaning up deployment history files on page 2133.
Days to Retain Deployment History	Enter the number of days to retain deployment history. This setting can be overridden with the related environment setting.
Enable Generic Process History Cleanup	When enabled, the generic process history files are set to delete as per the **Days to Retain Generic Process History** setting.
Days to Retain Generic Process History	Enter the number of days to retain generic process history. This setting can be overridden with the related environment setting.
Enable Process Step Log Cleanup	When enabled, the process step log cleanup starts. The process step logs are deleted according to the values that you set without deleting other history records.
Days to Retain Process Step Logs	Enter the default number of days to retain process step logs for generic processes not associated with deployments, and for environments if the number of days is not specified directly in the environment settings. Valid range is 0 or greater.

Table 687. Audit Log settings

Field	Description
Enable Audit Log Entries for Read Events	This option is disabled by default on new installation. When disabled, no READ events are recorded in the audit log.
Enable Audit Log Cleanup	When enabled, audit log entries older than the number of days specified in the **Days to Retain Audit Log Entries** field are removed beginning at the time specified by the **Daily Cleanup Start Time** field. See Configuring log rotation settings on page 3002. Deleting audit logs from **Settings > Audit Log > Cleanup Log** creates a log entry of event type Audit Log Cleanup that are non-deletable.
Daily Cleanup Start Time	If **Enable Audit Log Cleanup** is enabled, specify the time the audit log cleanup starts. The default value is 12:00 AM.
Days to Retain Audit Log Entries	If **Enable Audit Log Cleanup** is enabled, specify the number of days that audit log records are retained. The default value is 30.

Table 688. Password requirement settings

Field	Description
Minimum Password Length*	Define the minimum length for passwords.
Require Uppercase Characters in Passwords	When enabled, passwords must contain at least one uppercase character.
Require Lowercase Characters in Passwords	When enabled, passwords must contain at least one lowercase character.
Require Special Characters in Passwords	When enabled, passwords must contain at least one special character.
Require Number Characters in Passwords	When enabled, passwords must contain at least one numeric character.
Require Password to be Unique From User Information	When enabled, passwords must not match username, actual name, or configured email.

Table 689. Security settings

Field	Description
Discovery Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Auto Discovery Steps to expire. The value must be 300 seconds or more.

Table 689. Security settings (continued)

Field	Description
Configure Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Auto Configure Steps to expire. The value must be 300 seconds or more.
Version Import Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Version Import Steps to expire. The value must be 300 seconds or more.
Plugin Step Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for plug-in steps to expire. The value must be 300 seconds or more.
Use Default Auth Token Restriction if not specified	When enabled, the default authentication token restriction for processes that are created in Deploy is used. For these versions, the default token is applied to the most current process version but not other versions. For more information, see Restricting authentication tokens on page 2223.
Require Token Restriction	When enabled, the None option is removed from the Auth Token Restriction drop-down when creating or editing an authentication token. This makes Auth Token Restriction a required field, but the user can still set a highly permissive token restriction.
"Remember me" duration (hours)	Specify the number of hours a user can optionally be remembered and remain logged in.
Maximum Login time (hours)	Define the maximum duration (in hours) a user can stay logged in. When set, it forcibly disconnects the remote users from server when the logon hours expires. This setting overrides the **"Remember me" duration (hours)** setting. You can enter `0` or a negative value to disable the setting.
Logout should invalidate all "Remember Me" cookies	When enabled, invalidates all "Remember Me" cookies for a user when they explicitly log out.
Default Teams for New Objects	Select **All Teams Available to User** to add all teams that are available to user by default when a new object is created. When set to None, no teams are assigned to the users when new objects are created. Users can override this setting while creating objects.

Table 689. Security settings (continued)

Field	Description
Trusted proxy list	Enter a regular expression pattern to match IP of the trusted proxy servers. This value affects all servers. You must restart the servers for the changes to reflect. If left blank, the proxy headers are not trusted from incoming connections, which further may affect auth token IP restriction validation and IP logging.
User Idle Days to Lock	Specify the number of days after the user's last login date before locking the user account. You can enter 0 to disable user account locking.
User Idle Days to Delete	Specify the number of days after the user's last login date before deleting the user account. You can enter 0 to disable user account deletion.
Preview Idle User Lock	Click **Preview Idle User Lock** to load the information about user accounts that will be locked or deleted.
Realm for Bearer Authentication	Select an authentication realm to be used for the bearer authentication.

Table 690. Mail server settings

Field	Description
Mail Server Host	Enter the host name of the mail server that is used for notifications. Deploy can send notifications to users based on user-configured trigger events (to set up notifications, see Creating Notifications in a Notification Scheme on page 2081). Deploy requires an external SMTP mail server to send messages. To disable notifications, leave the field blank.
Mail Server Port	Enter the SMTP port that is used by the notifications mail server.
Secure Mail Server Connection	When enabled, the SMTP server connection is secured.
Mail Server Sender Address	Enter the sender's email address for email notifications.
Mail Server Username	Enter the user name for sending email notifications.

Table 690. Mail server settings (continued)

Field	Description
	Some email servers and firewalls treat emails with different sender and user names suspiciously. Therefore, consider specifying the same name for both fields.
Mail Server Password	Enter the user password for sending email notifications.
Max Recipients Per Email	Specify the maximum number of recipients to include in a single email. Mutliple emails will be sent when this is exceeded.

Table 691. External Approval settings

Field	Description
Resource Type to Run External Approvals	Choose any of the following options: • None: to use no agent or agent pool to run external approval processes. • Agent: to use an agent as the default resource type to run external approvals. Specify agent in the **Agent for External Approvals** field which will be used for external approval processes. • Agent Pool: to use an agent pool to specify a group of agents to run external approvals on instead of using a specific agent. Specify agent pool in the **Agent Pool for External Approvals** field which will be used for external approval processes. For more information, see Managing external approval processes on page 2392.

Table 692. Legacy Configuration Options

Field	Description
Enable "Prompt On Use"	When enabled, the option to set Prompt On Use for properties in workflow steps is displayed.
Fail Processes With Unresolved Properties	When enabled, any running process which comes across a property which it cannot resolve will automatically fail.
Component Environment Properties Override Environment Properties	When enabled, component environment properties overrides environment properties when using `${p:propName}` notation as opposed to `${p:environment/propName}`.
Enable Deleting Resources From Environments	When enabled, allows you to delete resources that are mapped with environment while deleting that environment.

Table 692. Legacy Configuration Options (continued)

Field	Description
	Note that the resources are deleted globally and affects the other environments that uses the same resource tree.
Show Compliancies Automatically	When enabled, environment compliancies are loaded automatically on the **Application Environment** and **Snapshot** pages rather than loading on demand.

Setting your profile

As a HCL DevOps Deploy (Deploy) user, you can manage teams and roles for your account, and preferences such as display options, language for the server, and how the server shows times and dates.

Managing the teams and users for your account

You can view and manage the teams and roles that you are assigned on the server. Your permissions determine which changes you can make.

Setting the teams for your account

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > Team Manager**.
2. Manage the teams that you are a member of.
   For more information, see Security teams on page 2213.

Setting the users for your account

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > User Manager**.
2. Manage the user that you are a member of.
   For more information, see Security teams on page 2213.

Setting the user preferences

You can set the preferences such as language for the server, and how dates and times are shown on the server.

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > Preferences**.
2. Choose a language for the user interface in the **Locale** field.
3. Select a format for dates in the user interface in the **Date Format** field.
4. Select a format for times in the **Time Format** field.
5. Specify the calendar type you want to use for the server in the **Calendar** field.
6. Specify which teams to assign for new objects in the **Default Teams for New Objects** list.
   By default, **System Default Policy** is selected and uses the **Default Teams for New Objects** settings defined in the system settings.

- **All Available Teams** to add new objects to all of the teams that you are a member of.
- **Specific Teams** to specify specific teams, and then select one or more teams.
- **None** to specify that no new objects are added to your teams.

7. Click **Save**.

Agents and agent relay configuration

After you install an agent for HCL DevOps Deploy (Deploy), you can modify its configuration or create an agent relay to facilitate communication to remote relays.

To monitor the status of the relay and the agents that are connected to it, click **Resources > Agent Relays**. This tab shows the number of active and offline agents that are connected to each relay. For more information about an agent relay, in the **Name** column, click the relay.

Agent security and communication

Agents use SSL-secured WebSocket, and HTTPS protocols to communicate with the server.

Agent overview

Agents do the actual work of deployment, which removes the task from the server. Agents are an important part of scalability in HCL DevOps Deploy (Deploy).

Although an agent is typically considered as a single process, technically an agent consists of a *worker* process and a *monitor* process. The worker process is a multithreaded process that runs the deployment work after it receives commands from the server. Work commands come from plug-in steps, which provide integration with many third-party tools. The monitor is a service that manages the worker process: starting and stopping, handling restarts, upgrades, and security, for example. After an agent is installed, it can be managed from the Deploy web application.

Agent security

All processes, including packaging, configuration, and deploying, that the Deploy server requests run on hardware that is assigned to agents. For added security, agents do not listen to ports, but open direct connections to the server instead. After an installed agent starts, the agent opens a WebSocket connection to the Deploy server based on the installation information.

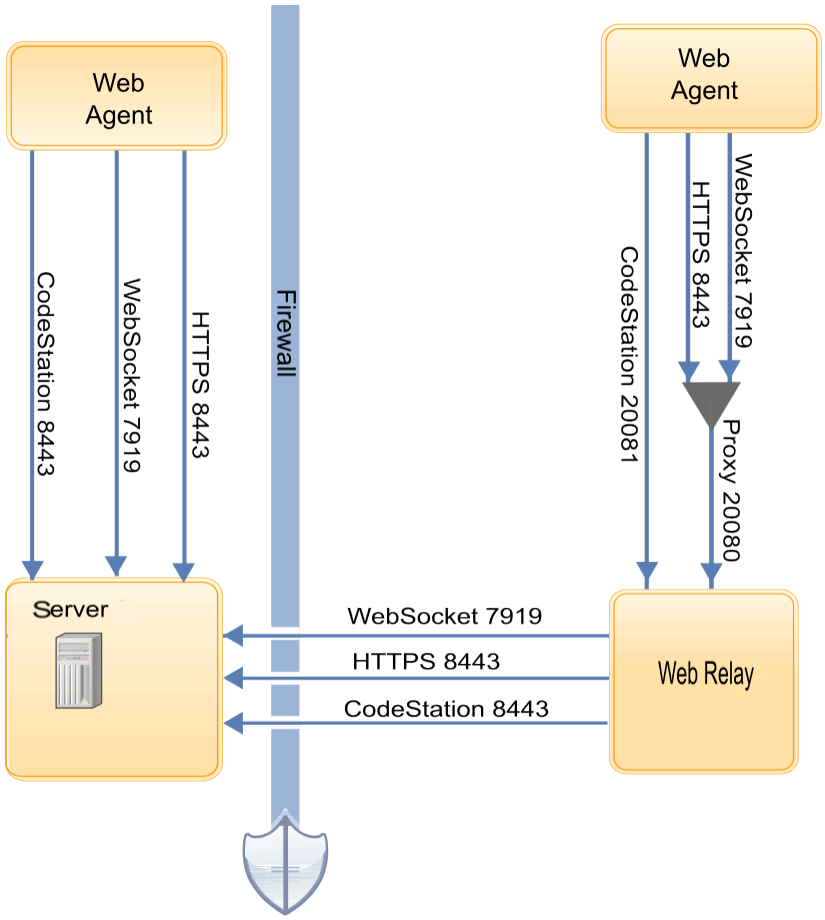
Agents on networks other than the one where the server is located might need to open a firewall to establish connection. After communication is established, the agent will be visible in the Deploy web application where it can be configured. Active agents, regardless of operating system, can be upgraded with the web application.

Agent communication

Agents are installed as Web agents on page 2096. Web agents use WebSocket connections and HTTPS for agent-server communication.

Web agents uses WebSocket connections for tracking agent status and notifications, and HTTPS for everything else.

The following diagram shows the main default ports that are involved in communication between agents, agent relays, and the server.



Stateless server-agent communication provides significant benefits to performance, security, availability, and disaster recovery. Because each agent request is self-contained, a transaction consists of an independent message, which can be synchronized to auxiliary storage as it occurs. Either endpoint, server or agent, can be stopped and restarted without repercussion, other than lost time. If communications fail midtransaction, no messages are lost. After they are reconnected, the server and agent automatically determine which messages got through and what work was completed. After an outage, the system synchronizes the endpoints and recovers affected processes. The results of work that an agent completes during the outage are communicated to the server.

Web agent configuration

Web agents use WebSocket and HTTPS connections for communication. Web agents do not use JMS.

Introduced with HCL DevOps Deploy (Deploy) V7.0.0, web agents are more scalable and secure than the previous agent type. The earlier type is called the JMS agent.

Web agents use the WebSocket protocol which is a bi-directional, point-to-point message protocol. WebSocket connections start as HTTPS requests and upgrade to the WebSocket protocol.

In the product user interface, web agents are recognized by the **Web** designation. On the **Agents** page, for example, web agents are the **Web** type.

Web agent communication

For successful server and web agent communication, you must have configurations on the server and agent sides.

In most cases, the installer configures the properties that are mentioned later. This section provides the additional information that you must consider if issues arise in server and agent communication.

Server configuration

During the server installation, you specify a secure WebSocket connection. The connection information has the following format: `agentcomm.uri=wss://physical-host:port`. The information is stored in the `server/conf/server/installed.properties` file. The default port is 7919. This port serves WebSocket endpoints that servers and web agents use and a REST API that only web agents use. While setting the `agentcomm.uri` property, make sure it meets these requirements:

- The URL is in this format: `wss://physical-host-or-ip:port`. The `wss` URL scheme is required and is the only supported scheme. The port must be explicit.
- The URL port must not conflict with ports for the UI.
- Each server must have a unique URL. URLs that refer to `localhost`, a loopback address, or a load balancer only work for a single server configuration.
- Every server in an HA cluster must be able to connect directly to this server with the URL.
- Agents and relays must be able to connect to one or more servers at the servers' URL ports. They can optionally be configured to use a common URL that all servers share, for example, one provided by a load balancer. However, servers must be configured with direct URLs.
- Relays support proxying HTTPS requests only. Any request through the relay that try to do HTTP are rejected with 501 error code. However, the relay itself listens on an HTTP protocol.

When you use web agents, each server in a cluster has a distinct URI and each server establishes a direct connection to every other server, which results in two connections between each server. The server-to-server connections advertise connected web agents to the cluster and exchange messages for them.

When a new server is added to a Web agent-enabled cluster, it connects to the previously defined servers. The other servers respond by connecting back to the new one. In normal operation, each server has two connections to each of its peers.

Web agent configuration

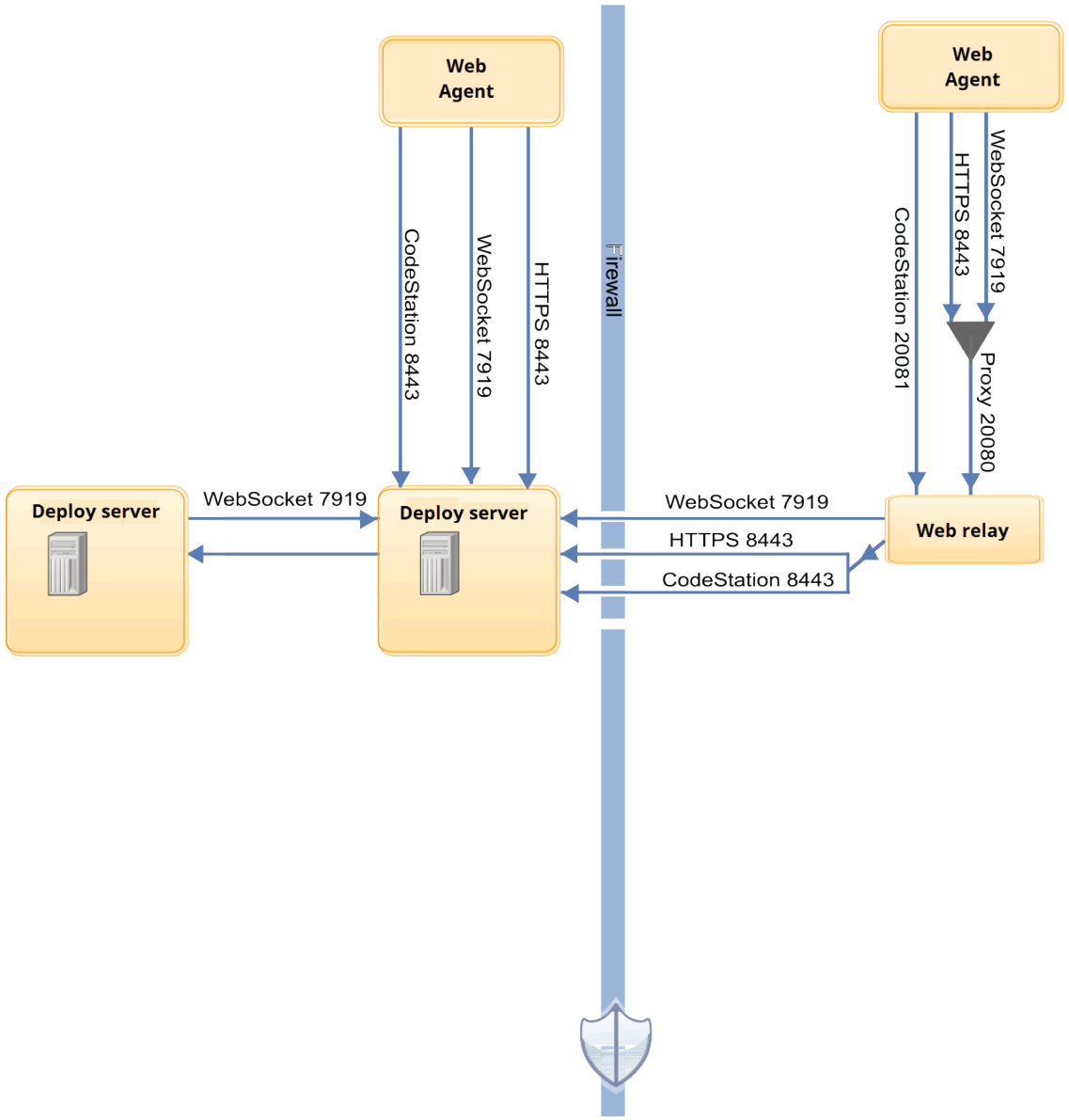
When you install a web agent, you specify the `agentcomm.server.uri` property. If there is one server or if all servers share a URL, it can be configured directly, for example, `agentcomm.server.uri=wss://examplehost.org:7919`. Configure multiple servers as a comma-separated list that is embedded in a random scheme, for example, `agentcomm.server.uri=random:(wss://examplehost1.org:7919,wss://examplehost2.org:7919)`.

Agents and relays connect to one or more servers at their respective URL ports. They can be configured to use a common URL, for example, a load balancer. However, servers must be configured with direct URLs. The `agentcomm.proxy.uri` property

can define a single HTTP proxy URL or define a list in a random scheme. The property is optional if the agent connects directly to servers.

Note:

The following diagram shows the default ports that are involved in communication between web agents, web agent relays, and servers.



Web agent relay configuration

Relays can be configured to support web agents exclusively.

You must upgrade relays for agents with web communication enabled and associate with the relay from the UI. However, a successful agent communication does not require any configuration. The following web communication parameters are defined in the *relayInstallDir*/conf/agentrelay.properties file:

```
agentcomm.enabled=true
agentcomm.server.uri=random:(wss://deploy1.example.org:7919,wss://deploy2.example.org:7919)
```

Agent relay limits

The maximum number of agents that can be assigned to a relay is limited in each of the releases. See to the following table to check the number of agents that are allowed for a relay for your installedHCL DevOps Deploy (Deploy) version.

Version	Relay mode	Agent type	Limit with default proxy
7.0.4.0 to 7.0.5.0	Web mode	Web Agents	1K
7.0.5.0 or later	Web mode	Web Agents	10K

In Linux, use the following utilities to reach the design limit of 10K concurrent agents:

- sysctl:

  ```
  fs.file-max >= 40000
  ```

- ulimit:

  ```
  nofile >= 40000
  ```

- sysctl:

  ```
  net.ipv4.ip_local_port_range >= "10000 65000"
  ```

Managing agents remotely

After an agent is installed, you can manage (with the monitor process) many of its features from the HCL DevOps Deploy (Deploy) web application.

About this task

We characterize an agent as a single process, but technically an agent consists of two processes: a *worker* process and a *monitor* process. Worker processes do the actual work of deployment, such as handling plug-in steps. Monitor processes manage the worker process, such as handling restarts, upgrades, and tests. Agent properties can be changed directly by editing the agent's conf/agent/installed.properties file and restarting the agent.

Note: Agents can access Windows shared drives. Drives are mapped on a per-user basis, you can check if the agent has the drive mapped with *fsutil fsinfo* drives.

1. Display the **Agents** pane (**Resources > Agents**).
2. Click **Actions** button on the desired agent.
   Actions are described in the following table.

Table 693. Agent management

Action	Description
Edit	Edits the agent name, description, and team configuration.
Delete	Deletes the agent permanently from the system.
Mark as Unavailable	Makes the online agent unavailable for running jobs or being used by agent pools. The `Unavailable` status acts like `Offline` status without having the agent or agent service shut down. You can make the unavailable agents online by using the **Mark as Available** option from the actions menu.
Restart	Shuts down and restarts the agent. While the agent is shut down, its status is `Offline`.
Test Connection	Runs the agent settings and connection test. Test results are displayed in the **Connection Test** dialog.
License	Sets the license from `unlicensed` to `authorized`.
Set License Type	Changes the license type.
Set Configuration Template	Applies an agent configuration template to the agent.
Upgrade	This option shuts down the agent and applies the upgrade. While the agent is shut down, its status is `Offline`. After the upgrade is applied, the agent is restarted.
Set Java Home	Updates JAVA_HOME for the agent. The Java Home lists the packages that are installed on the agent. Select an appropriate package or enter the absolute path of a valid JAVA_HOME directory.
Install Java Package	Installs the Java package on the agent. The Java Package list displays the packages that are added to the system and matches the agent architecture and OS.
Uninstall Java Package	Removes the Java package from the agent. The Java Package list displays the packages that are installed on the agent.

Note:

- The Java packages are added or managed in the **System Settings**. Ask your administrator, if you want a specific Java package added to the system.
- Agent properties and Java home information is available from the agent's **Metadata** tab.

Result

3. **Optional:** Perform the following actions on multiple selected agents using the **Bulk Actions** option:

Bulk action	Description
Restart	Shuts down and restarts the selected agents. While the agents shut down, their status is shown `Offline`. When the agents start, the status is changed to `online`.
Upgrade	Shuts down the selected agents and applies the upgrade. While the agents shut down, their status is shown `Offline`. After the upgrade is applied, the agents are restarted.
Add Tag	Allows you to apply tag to selected agents.
Remove Tag	Removes the applied tag from the selected agents.
Set License Type	Changes the license type for the selected agents.
Mark as Unavailable	Makes the online agents unavailable for running jobs or being used by agent pools.
Mark as Available	Makes the unavailable agents online for running jobs or being used by agent pools.
Apply Configuration Template	Applies the agent configuration template to the selected agents. The configuration templates must already be set the agents. See Migrating agents to Web on page 2107 for more information.
Set Configuration Template	Adds the selected agents to an agent configuration template.
Delete	Deletes the selected agents permanently from the system.
Install Preferred Java Package	Installs the Java package that are set as preferred in the Systems Settings. See Managing Java packages on agents in bulk on page 2103 for more information.
Set Java Home to Preferred Java Package	Updates the Java home of the selected agents as configured for their preferred Java package.
Remove Unused Java Packages	Removes Java packages for the selected agents that are neither active nor configured for the agent.

What to do next

Verify all actions and their details from the agent's **Maintenance** tab.

Related reference

Upload a Java package.

Install a Java package as a new managed Java installation

Set the configured JAVA_HOME of an agent

Restart an agent

Managing Java packages on an agent

You can upload, install, and manage Java packages on compact agent using the web UI.

Before you begin

Ensure that a supported Java package is downloaded to the system.

Important:

- See Operating systems for a list of supported versions.
- You can upload and manage Java only on the operating systems that have compact agents. Agents on operating systems like z/OS cannot manage Java.
- Deploy does not support native operating package formats or package installation tools.
- The compatible file extensions are `.zip`, `.tar`, `.tar.gz`, `.tgz`, `.tar.bz2`, and `.tbz2`.
- Newer IBM Java packages might have operating system prerequisites, such as installing updated system C++ runtime libraries. Deploy does not support installing such prerequisites.

About this task

You must upgrade Java when you have unsupported Java versions on your agents.

HCL DevOps Deploy (Deploy) identifies which agents must have an updated Java version or patches applied, and determines that status by using the real-time status of the connected agents. Java packages can be uploaded and then applied on different devices and architectures in your environment.

1. Upload the required Java package.
    a. Click **Settings > System > Manage Java Packages**.
    b. Upload a Java package by clicking **Add Package**.
    c. Provide a name for the Java package to upload.
    d. Click **Choose File**, select the Java package to upload and click **Submit**.
    The Java package is uploaded into the system.
2. Install the uploaded Java package on an agent.
    a. Click **Resources > Agents > *agent_name***.
    b. Click the menu option on the right of the agent for which you want to install the uploaded Java package.
    c. Select **Install Java Package** and specify the Java package name.
    d. Click **Install**.
    Result
    A message indicates that the Java package is being installed on the agent. When the installation finishes, the agent's Maintenance page displays the installed Java package information.
3. Go to the **Agents** page and set JAVA_HOME on the agent by selecting the **Set Java Home** menu option.

This location must be an absolute path, which is accessible to the agent.

Result

The agent restarts and runs with your new Java version.

Results

You have managed the Java packages on your agent.

What to do next

- See all Java versions installed for that agent from the agent's Metadata page.
- Remove unsupported Java package versions from the **Uninstall Java Package** menu option.
- If required, you can download this Java package at any point of time from the **Settings > System > Manage Java Packages** page.

Related reference

Upload a Java package.

Install a Java package as a new managed Java installation

Set the configured JAVA_HOME of an agent

Restart an agent

Download available Java package.

Managing Java packages on agents in bulk

You can manage the Java distribution on your numerous agents by using **Bulk Actions**.

Before you begin

Ensure that a supported Java package is downloaded to the system.

Important:

- See Operating systems for a list of supported versions.
- You can upload and manage Java only on the operating systems that have compact agents. Agents on operating systems like z/OS cannot manage Java.
- Deploy does not support native operating package formats or package installation tools.
- The compatible file extensions are `.zip`, `.tar`, `.tar.gz`, `.tgz`, `.tar.bz2`, and `.tbz2`.
- Newer IBM Java packages might have operating system prerequisites, such as installing updated system C++ runtime libraries. Deploy does not support installing such prerequisites.

About this task

When you have multiple agents running in your production environment, it can become difficult to apply Java patches regularly. Using **Bulk Actions**, you can manage Java packages on your agents with just a few clicks.

You must consider the following scenarios before you manage Java on your agents:

- Deploy does not support native operating package formats or package installation tools.
- The compatible file extensions are `.zip`, `.tar`, `.tar.gz`, `.tgz`, `.tar.bz2`, and `.tbz2`.
- Newer IBM Java packages might have operating system prerequisites, such as installing updated system C++ runtime libraries. Deploy does not support installing such prerequisites.

1. Upload the required Java package.
    a. Click **Settings > System > Manage Java Packages**.
    b. Upload a Java package by clicking **Add Package**.
    c. Provide a name for the Java package to upload.
    d. Click **Choose File**, select the Java package to upload and click **Submit**.
    The Java package is uploaded into the system.
2. Set the Java package as preferred for the OS and CPU type from the menu option.
    Typically, you must set at least one Java package as preferred for each OS and CPU type.
3. Go to the **Agents** page and select the agents for which you want to update the Java.
4. Install the preferred Java package by clicking **Bulk Actions > Install Preferred Java Package** and confirming the action.
    A message window opens, indicating that the Java installation has started. The installation might take some time. You can close the message window while the installation runs in the background.
5. Verify that the Java is installed for the agents by checking the **Version** column.
6. Set the JAVA_HOME on the agents by clicking **Bulk Actions > Set Java Home to Preferred Java Package** and confirming the action.
    A message window opens, indicating that the Java update request was sent. This step might take some time. You can close the message window while the JAVA_HOME is updated in the background.
7. Verify that the Java home is updated for the agents.
8. Restart the agents by clicking **Bulk Actions > Restart**.
    Agents restart and run with the new Java version. You can refresh the **Agents** page to check if the agents are online.

Results

You have managed to upload, install, and update Java packages on your agents in bulk.

What to do next

You can remove unused Java package versions by clicking **Bulk Actions > Remove Unused Java Packages**.

Related reference

Upload a Java package.

Set a package as preferred for its OS and CPU

Install Java packages as a new managed Java installations in bulk

Set the configured JAVA_HOME of agents in bulk

Updating the JRE location for agents

If you upgrade the JRE and change the JRE location on agents, you must also update the JRE location in the agent Java configuration file.

Before you begin

- Verify the supported Java version for agents. See Prerequisite software.
- You have installed the new JRE on the target agent and set the JAVA_HOME environment variable of the system.

Important: Do not use the server UI to restart the agent.

1. On the agent system, stop the agent.
   See Agent executable command-line commands on page 2339.
2. Edit the location of the JRE in the Java home configuration file.
   - On Linux™ or AIX, edit the *<DEPLOY_AGENT_HOME>*`/conf/agent/javahome` file.
   - On Windows™, edit the *<DEPLOY_AGENT_HOME>*`\conf\agent\javahome.conf` file.

     Note: For example, add the new location of the JRE on Windows™ as:

     ```
     C:\Program Files\Java\jdk-17.0.0
     ```

   - If the agent is running as a service on Windows™, remove and reinstall the service. See Removing and reinstalling agents as Windows services on page 2109.

     Note: For the Windows service of agents, you must update the **JAVA_HOME** property in *`installation_directory`*`\conf\agent\javahome.conf` file with the new location of Java, and then recreate the Windows service.
3. On the agent system, start the agent.
   See Agent executable command-line commands on page 2339.

Results

You have updated the JRE of the active agents.

Creating and managing agent pools

Similar to resource groups, *agent pools* help you organize and manage agents that are installed in different environments.

You can use an agent pool to spread the deployment processing work among participating agents and provide relief to otherwise overburdened agents. Users assign agents to pools and pools are assigned to resources, just like lone agents. When an agent pool is assigned to a resource, work items are sent to eligible agents. If you don't use a pool, then the agent that is assigned to the resource is assigned all the processing work regardless of its current workload. Such a workload might lead to bottlenecks.

Note: The agent pool algorithm will assign the workload to the most available agents in the pool. Most available agents are determined by which agents have the minimum running plugin steps in the agent pool. If there are multiple agents in the pool with the same plugin step count, the agents the database returns first will be assigned the workload.

Creating an agent pool

To create an agent pool, specify the agents that are the members of the pool.

1. From the **Resources** tab, click **Agent Pools.**
2. From the **Agent Pools** pane, click **Create Agent Pool**.
   Result
   The **Create Agent Pool** window opens.
3. Specify the pool name.
4. **Optional:** Enter a description in the **Description** field.
5. **Optional:** To manage the teams that can access the agent pool, click the plus icon by **Teams**.
6. To add agents to the pool, click the **Add Agents** field.
   Result
   A list of the available agents appears.
7. Select the agent or agents to add to the pool.
   You can filter the listed agents by entering search text into the field.
8. When you are finished, click **Save**.

Managing agent pools

You can manage agent pools from the web interface.

1. To display the **Agent Pools** pane, click **Resources > Agent Pools**.
2. Click an action link for the pool.
   Actions are described in the following table.

Action	Description
Edit	With this option, you can add or remove agents and edit the pool's name and description.

Action	Description
Copy	Creates a pool with the same agents as the selected pool.
Delete	Removes the agent pool.

To deploy a resource to an agent pool, the resource must be a child of the agent pool.

Agent pools for containers

Agent pools can be used to provide efficiency with your use of Deploy licenses. You can take advantage of running your agents in a Kubernetes-based environment where you can spin agent instances up and down quickly.

Follow these steps in order to achieve this efficiency:

1. Build the application resource tree using an Agent Pool instead of using the Agent directly. .
2. Deploy an instance of the Deploy Agent to your Kubernetes cluster. See Deploying Agent in Kubernetes Cluster.
3. The agent will be automatically get a license per the system setting on licensing agent.
4. Use the REST API to add the agent into the specified Agent Pool. See Add an agent to an agent pool.
5. Run your deployment.
6. Use the REST API to remove the agent from the specified Agent Pool. See Remove an agent from an agent pool.
7. The license will be removed if you are using server-agent license model; license will be released 15 minutes later if you are using Simultaneous Session license model.
8. Delete the instance of the Deploy Agent from the Kubernetes cluster.

Migrating agents to Web

Agent Configuration Templates allow you to migrate JMS agents on a server to Web agents with the ability to revert back to the original JMS configuration. This allows you to specify connection details for groups of agents.

Prerequisites

You must upgrade agents and server to HCL DevOps Deploy (Deploy) 7.0.1. For more information see Upgrading agents and Upgrading the server.

About this task

Deploy 7.0 introduced a new server/agent communication protocol which offers many performance improvements, and increased scalability. See Web agent configuration on page 2096. Deploy 7.0.1 is extending this capability by adding Agent Configuration Templates which will allow you to define a reusable set of configuration that can be used to easily migrate existing JMS agents to WEB agents. After you migrate JMS agents to WEB agents you have the ability revert them back to the original JMS configuration.

Note: You can only revert back an agent to JMS if that agent was previously a JMS agent.

Agent Configuration Templates

Before you can migrate an agent from JMS to Web an Agent Configuration Template will need to be assigned to the agent or agents.

1. Click **Resources** >**Agent Configuration Templates**.
2. Click on **Create Agent Configuration Template** on the right side of the page.
3. Give the template a name and description in the designated boxes.
4. Add teams that you want to give access.
5. Enter the URLs of the servers or relays where you would like agents to connect.
6. Click **Save**.
7. Click on the template name and click **Add Agents** to apply agents to the specific template.

After agents are added to templates they can be migrated to the new Web type.

Agent Migration

An Agent Configuration Template will need to be assigned to an agent before its type can be changed from JMS to Web.

1. Click **Resources** >**Agents**.
2. Select the agents that you want to convert.
3. Click bulk **Actions** menu.
4. From the **Actions** drop down dialog you can select **Migrate to Web** or **Revert to JMS**.

   Note: Agents can only be reverted to JMS if they were originally JMS.

If an agent is currently on a template there will be an icon next to **Type** on the **Agents** page. Clicking that icon will also prompt for agent migration.

A confirmation message will appear when the migration was successfully started on all agents.

The connection setting on newly migrated agents will be taken from the Agent Configuration Template that the agents belong to.

When migrating to web mode, the agents will perform a connection test that must pass for the migration to succeed. You can configure the threshold for the test to pass by providing an integer greater than one that specifies how many of the possible connection paths that have been configured.

Note: If migration fails you can view the logs by clicking on the agent name and then clicking on the **Logs** section of specified agent.

Removing and reinstalling agents as Windows™ services

If you installed an agent as a Windows™ service, you can remove and reinstall the service.

Old or non-compact agents

Before you begin, make sure that the agent is installed as a Windows service. See Installing agents from the command line for detailed instructions.

1. Open a command prompt as an administrator.
2. Go to the `C:\agent_install\bin\service` folder, where *agent_install* is the installed agent location.
3. Remove the service by running the following command:

   ```
   _agent.cmd remove service_name
   ```

   Use the name of the service for *service_name*. The default service name is `DevOps Deploy Agent`.
4. Reinstall the service by running the following command:

   ```
   _agent.cmd install service_name
   ```

   Note: While re-installing an agent, the latest PATH variable values are used.

Compact agents

Before you begin, make sure that the agent is installed as a Windows service. See Installing agents from the command line for detailed instructions.

1. Open a command prompt as an administrator.
2. Go to the `C:\agent_install\bin\` folder, where *agent_install* is the installed agent location.
3. Remove the service by running the following command:

   ```
   agent.exe uninstall-service
   ```
4. Reinstall the service by running the following command:

   ```
   agent.exe install-service
   ```

   For additional commands, see Agent executable command-line commands on page 2339.

Setting agent properties

Agent properties store information that are relevant to a specific agent.

About this task

Agents have many properties. These properties are categorized into:

- **Agent properties** - Agent properties are uploaded by the agent during installation and are stored in the database property sheet. These properties are only updated when the agent connects with the server and by using the agent property collection APIs.
- **User-defined properties** - User-defined properties are stored in VC and versioned when the properties are configured.

To view or set agent properties:

1. Navigate to **Resources > Agents >** ***agent name*** **> Configuration**.
2. View the non-versioned properties from the **Agent Properties** table.
3. View or update the versioned properties from the **User Defined Properties** table.
   You can click **Add Property** to add a property to the table. Each property can be deleted and edited.

Updating the JRE location for agent relays

If you upgrade the JRE and change the JRE location, you must also update the agent relays. If you do not change the location of the JRE, you do not have to update these files.

1. Stop the agent relay.
2. In the installation folder, locate and open this file for editing: *`install dir`*`/bin/agentrelay.cmd` (Windows) or *`install dir`*`/bin/agentrelay` (Linux).
3. In the file, edit the line that begins `set JAVA_HOME=` to point to your JRE.
   Example
   For example, `set JAVA_HOME=C:\Program Files\Java\jre8` (Windows) or `set JAVA_HOME=/opt/java/jre17` (Linux).
4. If you are running agent relays as a Windows™ service, follow these steps:
   a. Locate and open this file for editing: *`install dir`*`/services/agentrelay.cmd`.
   b. In the `agentrelay.cmd` file, edit the line that begins `set JAVA_HOME=` to point to your JRE.
      Example
      For example, `set JAVA_HOME=C:\Program Files\Java\jre8`.
   c. Open a command prompt as an administrator.
   d. Go to the *`agentrelay_install`*`\bin\service` folder.
   e. Remove the service.
      ```
      agentrelay.cmd remove service_name
      ```
      To find the service name, open the services view, and find the service with the following format: *`service_name`* `Agent Relay`.
5. Start the agent relay.

Changing passwords for the agent relay keystores

You can change the Java™ keystore passwords for the agent relay.

About this task

Each agent relay contains two keystores. The **agentrelay** keystore controls the JMS connection between the agent relay and the server and agents. The **codestation** keystore controls HTTPS connections to the agent relay and is used with server identity verification. See Enabling server identity verification on page 2152.

The password for each agent relay Java™ keystore must be identical to its certificate password. After you change both the keystore password and the corresponding certificate password, you must update the password in the agent relay properties.

Changing the passwords requires the keytool application, which is included in the Java™ developer kit and is not part of HCL DevOps Deploy (Deploy).

1. Stop the agent relay.
2. Change the **agentrelay** keystore password for the agent relay.

    a. Open a command-line window, and go to the `/agent-relay-install/conf/jms-relay` directory.
    By default, the *agent-relay-install* location is `/opt/devops-deploy/agentrelay` on Linux™ and `C:\Program Files\devops-deploy\agentrelay` on Windows™.

    b. Change the **agentrelay** keystore password by using this command:

    ```
    keytool -storepasswd -new newpassword -keystore agentrelay.keystore
    -storepass changeit
    ```

    The default password for this keystore is *changeit*.

    c. Change the **agentrelay** certificate by using this command:

    ```
    keytool -keypasswd -alias ucd-agentrelay-id -keypass changeit -new newpassword -keystore
     agentrelay.keystore -storepass newpassword
    ```

    The default keystore alias is *ucd-agentrelay-id*.

    d. In a text editor, open the `agent-relay-install/conf/agentrelay.properties` file.

    e. Specify the new Java™ keystore password for the agent relay in the following line of code:

    ```
    agentrelay.keystore.password=newPassword
    ```

3. Change the **codestation** keystore password for the agent relay.

    a. Open a command-line window, and go to the `/agent-relay-install/conf` directory.
    By default, the *agent-relay-install* location is `/opt/devops-deploy/agentrelay` on Linux™ and `C:\Program Files\devops-deploy\agentrelay` on Windows™.

    b. Change the **codestation** keystore password by using this command:

    ```
    keytool -storepasswd -new newpassword -keystore codestation.keystore
    -storepass changeit
    ```

    The default password for this keystore is *changeit*.

    c. Change the **codestation** certificate password by using this command:

```
keytool -keypasswd -alias agentrelay -keypass changeit -new newpassword -keystore
 codestation.keystore -storepass newpassword
```

The default keystore alias is *agentrelay*, and its value is shown in the `codestation.cert.alias` property in the *`agent-relay-install`*`/conf/agentrelay.properties` file.

d. In a text editor, open the *`agent-relay-install`*`/conf/agentrelay.properties` file.

e. Specify the new Java™ keystore password for the agent relay in the following line of code:

```
codestation.keystore.password=newPassword
```

4. Start the agent relay.

Results

The agent relay keystore passwords are changed and are obfuscated in the *`agent-relay-install`*`conf\server\agentrelay.properties` file.

Managing agent relays remotely

After an agent relay is installed, you can manage many of its features from the HCL DevOps Deploy (Deploy) web application.

About this task

Starting with Deploy version 7.0.4, you can test, restart, and delete **Agent Relays** directly through the user interface.

1. Display the **Agents Relays** pane (**Resources > Agents Relays**).
2. Click the **...** next to the relay.
   Actions are described in the following table.

Table 694. Agent relay management

Action	Description
Restart	This option shuts down and restarts the agent relay. While the agent relay is shut down, its status is `Offline`.
Test	This option runs a connection test. Test results are displayed in the **Connection Test** dialog window.
Delete	Removes the agent relay.

Migrating agents relays to Web

JMS agent relays on a server can be migrated to Web with the ability to revert back to the original JMS configuration. This allows you to specify connection details for groups of agent relays.

Prerequisites

Agents, agents relay, and server must be upgraded to the HCL DevOps Deploy (Deploy) 7.0.4 or later. For more information, see Upgrading agents and Upgrading the server.

JMS agents are NOT supported on Web relays. Make sure that the connected agents are migrated to Web.

About this task

Up to version 7.1.0.0, migration of agent relays can be done manually by modifying the `agentcomm` properties in the agentrelay.properties file. Relays can be automatically migrated to Web while upgrading the agent relays. For more information, see Upgrade Agent Relays. This task explains the procedure to migrate the agent relays manually.

Note: You can only revert back an agent to JMS if that agent relay was previously a JMS relay.

Manual Migration

1. Stop the relay you want to migrate.
2. Open the *relayInstallation*`/conf/agentrelay.properties` file.
3. Edit the file as follows:

```
agentcomm.enabled=true
agentcomm.server.uri=random\:(wss\://host-or-ip\:7919)
```

   Note: Change `host-or-ip` to your Deploy server name.

4. Restart the relay.
5. Verify that the Web agents are connected to the Web relay.

Note: If migration fails, you can view the logs in the *relayInstallation*`/var/log` directory.

Licensing

HCL DevOps Deploy (Deploy) provides a 60-day license-free evaluation period. You must have a license to use all product features after the evaluation period.

Unless you provide the location of a license server when you install Deploy, a 60-day evaluation period begins when you install the product. During the evaluation period, you can use all product features. A message on the web-based UI displays the number of days that remain in this period.

After the evaluation period, you can access Deploy to view and edit items but you cannot run deployments. You need a license for the agents in which you want to run the deployments. You can apply for a license at any point in time.

Note: Relays do not require any additional licenses.

HCL licensing

HCLSoftware offers My HCLSoftware (MHS) as a software license manager. The MHS portal has utilities to acquire and return licenses for `Authorized` and `Authorized zOS` agents. The licensing capabilities of MHS also provide the two important details:

- Maximum number of authorized licenses
- Maximum number of authorized z/OS licenses

An organization identifies one of its resources as a License Manager (also called Tech or Portal Admin) who is familiar with the language of licenses. The license manager is responsible for following the instructions in the document to activate entitlement, create deployments, and download the software from the portal.

Note: For information about HCL licensing, refer to the HCL Licensing documentation.

Grace period

A grace period is a duration of time that starts when Deploy server is not able to connect with the license server. You have a 24-hour grace period to restore the lost connection. During the grace period:

- You can see the following message on the banner:

  License Server Connection Issue Detected
  The connection to the license server has been lost. Deployments will work for the next 24 hours. Please contact support.

- You cannot license or unlicense agents.
- You can run the deployments for the duration mentioned in the banner.
- Deploy server continues to poll the license server and tries to establish a connection with the license server.

After the grace period ends, the licensing system proceeds to unlicense the agents. To debug licensing issues, you can add the following logger to the Deploy server:

```
com.urbancode.ds.subsys.deploy.license.hcl.HclLicenseUtil
```

See Managing loggers on page 3000 for information about how to add loggers.

License types

Depending on your agents operating system and intended use case, you can decide the license type you might need for your HCL DevOps Deploy (Deploy) agents.

There are four types of licenses available for Deploy agents:

- Authorized license for typical (non-z/OS) agents
- Authorized license for z/OS agents
- Floating license for typical (non-z/OS) agents
- Floating license for z/OS agents

Note: The Deploy server does not require any licenses in order to connect to and interact with the Web UI.

Licenses are applied to agents via the Web UI. You can configure the server to automatically apply licenses to new agents as they connect. For information about how to apply licenses, see Applying HCL licenses to agents on page 2115.

Authorized license for typical agents

The authorized license type allows you to allocate license to specific agents until they are deleted or unlicensed.

The authorized type license for typical agents is shown as `Authorized (HCL_Agent)` in Deploy.

Authorized license for z/OS® agents

Agents on z/OS® use different licensing than the typical agents. However, the z/Linux™ agents use the same licensing as the typical agents. Each z/OS agent requires a license, even if multiple agents are running on a single logical partition (LPAR). One agent is required on each environment, even if the environments are on a single z/OS sysplex.

The authorized type license for z/OS agents is shown as `Authorized (HCL_Agent_Z)` in Deploy.

Floating license type for agents and z/OS® agents

Floating licenses are required when you want multiple agents to share a limited number of licenses. In floating license type, the licenses are not tied to agents. When an agent which is not licensed performs a deployment, the server looks for an available license from the license pool. If a license is available, it reserves the license for that agent until the work is done. When the agent completes its work, the license is returned to the license pool for other agents to work.

If there are insufficient licenses for the number of floating agents, then the server records that but it still allows deployment to continue.

The floating type license for typical agents is shown as `Floating (HCL_Floating_Agent)` and for z/OS agents is shown as `Floating (HCL_Floating_Agent_Z)` in Deploy.

Applying HCL licenses to agents

You can apply licenses to HCL DevOps Deploy (Deploy) agents to run deployments in your production environment.

Before you begin

- You must have an entitled agent license and must have acquired the HCL Local License Server (LLS) or HCL Cloud License Server (CLS) URL.
- Check if the license server is reachable by opening the license server URL in your browser or by using the following curl command:

  ```
  curl --cacert /path/to/your/certificate.crt https://FQDNhostname:LLSport/v1/licensepools/deploymentID
  ```

  You can check the number of allocated licenses and used licenses on a successful connection. Refer to the HCL Licensing documentation for more information.

About this task

As an administrator or a license manager, you can configure Deploy to use the entitled license.

1. Click **Settings > System > License Management** on the dashboard.
2. Set the **Automatic Agent License Type** field to automatically apply licenses to newly created agents.

   Note: The available license types are stated by the current server license mode. For example, set the **Automatic Agent License Type** as HCL_Agent for standard agents or HCL_Agent_Z for z/OS agents.

3. Enter the license server URL in the **License Server Path** field.
   The example URLs for your license type are as follows:
   - HCL Local License Server:

     ```
     https://lls-host:lls-port/v1/licensepools/deploymentID
     ```

   - HCL Cloud License Server:

     ```
     https://cloud_server/v1/licensepools/deploymentID
     ```

4. Click **Save Connection**.

Results

Deploy is configured with the HCL license server and displays the license status in the **License Management** page.

What to do next

After applying the license, you confirm that the license is active by clicking **Resources > Agents**. The licensed agents are listed and flagged as `true` in the **License** column. In case of floating licenses, the **License** column shows `true` only when the agent is in use for deployment.

You can also enable logging for licensing. See Managing the logs on page 3002. The enabled licensing log is displayed as `True` in the log table. You can find the licensing log information in the `var\log\licensing.out` file.

Tracking agent usage

You can track daily agent usage. Agent usage can provide a rough estimate of agent license use.

HCL DevOps Deploy (Deploy) records the number of agents online and the maximum number of agents deploying at a time. Both measurements are referred to as the agent high-water mark. The records are updated every 10 minutes and are aggregated at the end of the day.

Although the agent usage records do not correspond precisely to agent license use, they can provide useful estimates of agent license use. These estimates can be especially useful for installations of Deploy on z/Linux™. If you are using z/Linux, review your agent high-water mark totals quarterly. If you discover that you significantly or routinely exceed your licenses, contact your sales representative to ensure that your installation remains compliant.

The agent usage records can be accessed at the following REST API endpoint:

```
https://{hostname}:{port}/cli/agentCLI/usage
```

The REST endpoint returns a JSON object with both the highest number of agents online for the day, and the maximum number of agents deploying at a time. A typical JSON object returned by the command is shown in the following example: `{"onlineAgents":5,"agentsrunningDeployments":4}`.

Both measurements are referred to as the agent high-water mark. For information about the CLI command, see getHighWatermarks on page 2777. For information about the REST API method, see Return high watermarks for agent usage.

Configuring the AI settings

You can configure the AI settings to integrate an AI service, such as OpenAI, Azure OpenAI, WatsonX, and Ollama. The AI service is used by the Deploy users to troubleshoot deployment failures.

Before you begin

You must have ensured that the AI provider details, such as endpoint and API key are ready.

1. Go to **Settings > System > AI Settings** from the Web UI.
2. Provide the following AI provider details:

Field	Description
AI Provider	Select the AI provider name. The OpenAI, WatsonX, and Ollama providers are supported. The AI settings are different based on your selection.
OpenAI settings	
AI Provider Endpoint	Enter the OpenAI service endpoint. If you change the endpoint, you must update the API key as well.
AI Provider API Key	Enter the API key that is used to make the OpenAI API requests.
AI Provider Project ID	Enter the OpenAI project ID, if created.
AI Provider Organization	Specify the unique organization ID.
Model Name	Choose a model name. The values in the **Model Name** field are automatically pulled based on the AI endpoint, API key, and organization details that you provided.
Azure OpenAI settings	
AI Provider Endpoint	Enter the OpenAI service endpoint. If you change the endpoint, you must update the API key as well.
AI Provider API Key	Enter the API key that is used to make the OpenAI API requests.

Field	Description
Deployment Name	Enter the name of the Azure OpenAI deployment to use for chat requests.
WatsonX settings	
AI Provider Endpoint	Enter the WatsonX service endpoint. If you change the endpoint, you must update the API key as well.
AI Provider API Key	Enter the API key that is used to make the WatsonX API requests.
AI Provider Access Token Endpoint	Enter the access token that authorizes access to WatsonX endpoint.
AI Provider Project ID	Enter the WatsonX project ID, if created.
Model Name	Choose a model name. The values in the **Model Name** field are automatically pulled based on the WatsonX endpoint and API key details that you provided.
Ollama settings	
AI Provider Endpoint	Enter the Ollama service endpoint.
Model Name	Choose a model name. The values in the **Model Name** field are automatically pulled based on the Ollama endpoint details that you provided.

3. Save your changes.

Results

You have configured the AI settings.

What to do next

You can configure a user role to allow them to use AI, and let them perform the AI analysis on the failed deployments.

Related reference

Permission reference on page 2192

Related information

Roles and permissions on page 2187

Performing AI analysis on failed deployments on page 3014

Setting up the wizard for new users

HCL DevOps Deploy (Deploy) includes a wizard to help users become familiar with the procedures and steps in creating a deployment. Some configuration is required before the wizard can be used.

Before you begin

At least one agent must be installed and accessible to users to start and complete the introductory wizard.

About this task

The wizard is enabled with new installations of Deploy. If you are working with an upgrade, the **Welcome** tab that includes the push button to open the wizard must be enabled. To enable the **Welcome** tab, open the `installed.properties` file, and add this entry: `server.enable.welcome.tab=true`. By default, this file is in this folder: `/opt/install-dir/server/conf/server`.

Tip: You don't have to create a role to assign so that they can follow the wizard. You can assign users to the Automation Engineer role, which is included on the server during installation. This Automation Engineer role has all the permissions that are required to complete the introductory wizard. If you want to create a separate role for wizard users, complete this procedure.

1. Create a role with these permissions:
    - Create Application
    - Create Component
    - Create Environment
    - Create Resource
    - View Agent
2. Give this role access to at least one agent.
3. Assign this role to users so that they can start and follow the wizard.

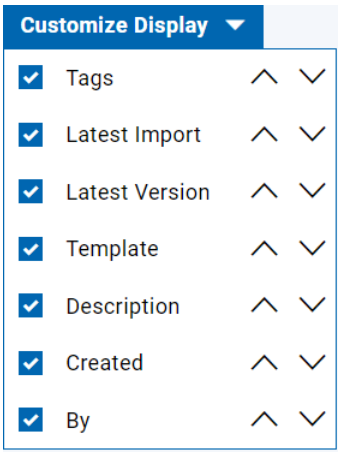
Customizing the table views

Customize a table view in the HCL DevOps Deploy (Deploy) application by hiding or showing, and reordering of the table columns.

About this task

You can customize table views available from the **Dashboard**, **Components**, **Applications**, **Processes**, **Resources**, and **Work Items** tabs using the **Customize Display** option.

1. Open the tab for which you want to customize the table view.
2. Click **Customize Display**.
   A list of table columns is displayed. The following screenshot displays an example of table columns:



By default, all columns are enabled.

3. Hide a column by deselecting the column name.
4. Reorder a column using the up and down arrow icons.

Results

The table view is updated as and when you make change.

Note:

- The changes are saved in a cookie and persist when you reopen the tab.
- At least one column is non-customizable. Generally, it is the **Name** column.

Managing data

This section gives details about how you can manage your data in HCL DevOps Deploy (Deploy)

Migrating Codestation from filesystem repositories to S3 repositories

Migrate your Codestation artifacts from filesystem to S3 using the S3 migrator tool.

About this task

You can migrate Codestation from filesystem into S3 repositories.

Table 695. Command line argument descriptions

Parameters	Description
bucketName	The name of the bucket in S3.
s3url	The URL of the S3 provider. Not required if using Amazon Web Services (AWS).

Table 695. Command line argument descriptions (continued)

Parameters	Description
s3region	The region of the S3 provider. You must specify either **s3url** or **s3region**, but not both.
s3user	The S3 username. Optional, depending on the S3 provider and system configuration. AWS uses APIKEY.
password	The S3 password. Optional. The value depends on the S3 provider and system configuration. AWS uses secret key.
signer	Optional. Consult S3 provider documentation for value. Not necessary for AWS.
enablePathStyleAccess	Optional. Consult S3 provider for documentation if this is necessary. Not necessary for AWS.
repositoryLocation	The location of Codestation repository. The default location in server is `appdata/var/repository`.
verify	Use this parameter to specify whether or not to enable verify mode. If you specify `true`, files are not uploaded to S3. Instead, the tool verifies that all expected files currently in Codestation on the filesystem are in S3.

Migrate Codestation from filesystem repositories to S3 repositories as follows:

1. Download **S3 Migrator Jar** file from the **Tools** page on machine that can access both current Codestation and S3 URLs.
2. Run the S3 Migrator tool.

```
java -jar s3migrator.jar -repo repositoryLocation -bucket bucketName [-url s3url] [-user s3user]
 [-password password] [-s3signerOverride signer] [-enablePathStyleAccess true|false(default false)]
 [-verify true|false(default false)]
```

Example usage:

```
java -jar s3migrator.jar -repo /opt/devops-deploy/server/appdata/var/repository -bucket
 deployCodestation
```

Note: The migrate tool will upload the files which are in Codestation into S3. This can be ran on a server other than the HCL DevOps Deploy (Deploy) server. It can also be ran while the Deploy server is up and handling requests. You can run the tool multiple times. Depending on the size of the Codestation repo, the process may take awhile. The tool will only upload files not currently in S3. This will allow the tool to be ran over

and over until the time to run it takes is short enough that the server can be shutdown for the last upload to ensure all artifacts are uploaded successfully.

3. Configure server to use S3 by setting the below properties based on the values you specified when you set up your S3 bucket.

```
  #installed.properties
  #required
codestation.provider=s3
codestation.s3.bucket=yourS3bucketname
codestation.s3.region=S3regionName or codestation.s3.url=S3customUrl

  #optional / setup dependent
codestation.s3.user=api_key
codestation.s3.signerOverride=signer
codestation.s3.enablePathStyleAccess=trueORfalse

  #secured-installed.properties
codestation.s3.password=api_secret
```

Note: Replace all user-provided values like *yourS3bucketname* with values as per your S3 configuration.

4. Restart the server.
5. **Optional:** If you need to roll back S3 migration to Codestation, run the migrator tool with updated properties. To recover artifacts from S3 repository, follow the external links below or contact support for assistance.
    - https://docs.aws.amazon.com/AmazonS3/latest/dev/replication.html
    - https://aws.amazon.com/backup-restore/
    - https://docs.aws.amazon.com/AmazonS3/latest/user-guide/undelete-objects.html
    - https://stackoverflow.com/questions/2426493/disaster-recovery-for-files-on-amazon-s3

Archiving version artifacts to Amazon S3

You can archive version artifacts to Amazon S3 to manage and prevent the accumulation of unwanted versions.

About this task

Configuring archiving to Amazon S3 involves adding S3 properties to the `devops-deploy\server\conf\server\installed.properties` file. After archiving is configured, the archived versions are stored in an Amazon S3 bucket in the specified region.

Note: Codestation artifacts and archived versions must use a different S3 bucket. If you configure them to use the same bucket name, the archiving does not work.

You can restore archived artifacts by navigating to **Components > *selected component* > Versions > *selected version*** and clicking **Restore from Backup** in the **Artifacts** table.

1. Configure the server to use Amazon S3 by adding or modifying the properties below:

```
# Example config
versionArchive.s3.bucket=yourS3bucketname
versionArchive.s3.url=S3Url
versionArchive.s3.accessKey=api_key
versionArchive.s3.secretKey=api_secret

# Alternative options
versionArchive.s3.region=S3regionName
versionArchive.s3.enablePathStyleAccess=trueORfalse
versionArchive.s3.signerOverride=SigV2

# Force on/off
versionArchive.s3.enabled=trueORfalse
```

Note:

- Replace all user-provided values such as *yourS3bucketname* with values according to your S3 configuration.
- Set `versionArchive.s3.enabled` to **true** to allow the server to check for the required configurations. If any configurations are missing, the server generates an error message. If set to **false**, missing configurations are ignored and S3 archiving is disabled.

If you have pre-configured Codestation S3 properties, you can reuse them by mapping them to the new version archive properties as follows:

```
# Codestation S3 config that can be reused
# Codestation Property (Source)       => Version Archive Property (Target)
codestation.s3.url                    => versionArchive.s3.url
codestation.s3.region                 => versionArchive.s3.region
codestation.s3.user                   => versionArchive.s3.accessKey
codestation.s3.password               => versionArchive.s3.secretKey
codestation.s3.enablePathStyleAccess  => versionArchive.s3.enablePathStyleAccess
codestation.s3.signerOverride         => versionArchive.s3.signerOverride
```

Optional: If you are using a RustFS container, you must configure the properties by specifying the following parameters:

```
# RustFS container config
# - enablePathStyleAccess=true is critical
# - do not put bucket name in url
versionArchive.s3.bucket=yourS3bucketname
versionArchive.s3.url=S3Url
versionArchive.s3.enablePathStyleAccess=true
versionArchive.s3.accessKey=api_key
versionArchive.s3.secretKey=api_secret
```

Note: You must enable the Path-Style Access, and the URL must not include the bucket name.

2. Save and restart the server.

Results

The archived artifacts are now zipped and moved to the specified S3 bucket.

Relocating CodeStation

You can move the CodeStation artifacts to a different location on the server.

About this task

By default, the server's log files and CodeStation artifacts are stored in the application data directory. You specified this directory when you installed the server. In the case of high-availability clusters, this directory is in a shared location where each server in the cluster can access it.

Ideally, this data is stored on robust network storage that is regularly synchronized with an off-site disaster recovery facility. In addition, the HCL DevOps Deploy (Deploy) server requires a fast network connection to this storage.

Tip: The following formula shows a good rule-of-thumb for determining CodeStation storage requirements: `average artifact size * number of versions that are imported per day * average number of days before cleanup`

Follow these steps to move the location of the CodeStation artifacts:

1. Stop the server.
   See Starting and stopping the server.
2. Move the CodeStation data to a new location.
   Which folders you move depend on whether the server uses an application data folder or not:
   - Check the `installed.properties` file on the server. If the **server.appdata.dir** property in this file has a value of two periods, (`..`) the server does not use an application data folder. If the property has a value other than two periods, the server uses an application data folder.

   When you know whether the server has an application data folder or not, you can move the correct files to a new location.

   Choose from:
   - If the server uses an application data folder, complete these steps:
     a. Move the application data folder to a new location. The default application data folder is `/opt/devops-deploy/server/appdata` on Linux™ and `C:\Program Files\devops-deploy\server\appdata` on Windows™.
     b. On the server, in the `installed.properties` file, update the **server.appdata.dir** property to the new location of the application data folder.

     If the new application data folder is on a remote system, you might have to set the **server.appdata.dir** property to a remotely mounted folder or other network storage.
   - If the server does not use an application data folder, complete these steps:
     a. Move the following individual folders to new locations:
        - *install_folder*`/var/plugins`
        - *install_folder*`/var/repository`
        - *install_folder*`/var/sa`

- *install_folder*/logs
- *install_folder*/conf/server/notification-templates
- *install_folder*/conf/encryption.keystore
- *install_folder*/conf/server.keystore
- *install_folder*/conf/collectors
- *install_folder*/patches
- *install_folder*/conf/server/log4j.properties

b. Create links from the previous folder locations to the new folder locations so that the server can continue to access the folders. For example, on Linux™ systems, you can use symbolic links, also known as *symlinks*.

3. Start the server.

Caching artifacts on agent relays

Agent relays can cache downloaded artifacts. Caching artifacts in this way improves performance because agents retrieve artifacts from agent relays instead of from the server.

When you enable artifact caching on an agent relay, the agent relay maintains a local CodeStation repository. This repository stores cached versions of artifacts that are on the server CodeStation repository. By default, the first time an agent requests an artifact, the agent relay retrieves the artifact, caches it in its CodeStation repository, and provides it to the agent. The next time an agent requests the artifact, the relay checks if the cached artifact is up to date. Then, the agent relay updates the cached artifact if necessary and provides the artifact to the agent.

By default, agent relays cache only artifacts that one or more agents request. You can also configure agent relays to pre-populate the cache with all artifacts with a specified status. In this case, the agent relays populate the cache with artifacts that have the specified status. Then, the artifacts are available immediately when agents request them.

Caching artifacts improves performance if the server is a long distance from the agents. In this case, caching artifacts on an agent relay that is close to the agents reduces the time that it takes to copy artifacts to the agents. Caching artifacts also improves performance if many agents retrieve artifacts from the server at the same time. In this case, caching the artifacts transfers server load to the agent relays.

Note: If you cache files on the relay, do not put the CodeStation cache on an NFS file system.

Before you begin

To pre-cache artifacts on agent relays, set up one or more component version statuses. Component versions with these statuses are cached on agent relays before agents request the component versions.

About this task

If you did not configure agent relay artifact caching when you installed the agent relay, you can also configure it by modifying its properties.

1. Open the */agent-relay-install*/conf/agentrelay.properties file.
2. Control caching with the `agentrelay.codestation.enable_replication` property:

Choose from:

- To enable caching, set the property to `TRUE`. An example true property resembles this code: `agentrelay.codestation.enable_replication=true`.
- To disable caching set the property to `FALSE`. An example false property resembles this code: `agentrelay.codestation.enable_replication=false`.

Note: If caching is enabled on an agent relay and disabled later, the cached artifacts are not deleted. To delete the artifacts manually, stop the agent relay and then delete the folders */agent-relay-install*/var/repository/ptr and */agent-relay-install*/var/repository/blob.

3. To preload artifact versions that contain certain statuses, specify the statuses in the `agentrelay.codestation.geotags` property.
   Separate statuses with commas. If you edit a status that contains a comma, the comma must be escaped.
   Example
   A modified `agentrelay.codestation.geotags` property might resemble the following code:
   `agentrelay.codestation.geotags=Hello%2c World,Production`

   Then, ensure that the component versions that you want to cache have one of these statuses.
4. Modify the cache size with the `agentrelay.codestation.max_cache_size` property.

Releasing VC LOB space

You can compress the amount of large object (LOB) storage space that Deploy uses in the `vc_persistent_record` database table.

Before you begin

Releasing the LOB storage space is optional. If you don't release the space, it is reused when new records are added. You can decide whether you want to release the LOB storage space for the following reasons:

- You can reduce storage space by 75%.

  Reduce the size of one of the largest tables in Deploy where the text data decreases by 50-75%.
  - A 2-byte character to an average of 1-byte per UTF-8 character which is up to 50% reduction.
  - XML might compress by 87% or more because XML data contains many repeated keywords.

- You can achieve up to a 50% reduction in read time.

Releasing LOB space in Oracle databases

You can release the large object (LOB) storage in the Oracle databases.

Before you begin

Plan downtime for upgrading HCL DevOps Deploy (Deploy) and releasing LOB space.

About this task

Run the following database commands to release LOB storage in Oracle databases. Releasing LOB storage in Oracle database requires different commands for **basicfile** and **securefile** file type.

You can also refer to the Oracle instructions for reclaiming unused space here: https://oracle-base.com/articles/misc/reclaiming-unused-space.

1. Set the **versioned_config.upgrade.enabled=true** parameter in the `conf/server/installed.properties` file.
2. Restart the Deploy server.
3. Check on the progress of the conversion:

   ```
   select count(*) from vc_persistent_record where persistent_data is not null
   ```

   The compression process is complete when the row count reported by the query is 0.
4. Check the database size:

   ```
   select sum(bytes)/1024/1024 size_in_mb from dba_segments;
   ```
5. Release LOB storage by running the following commands for the appropriate Oracle file type.
   - For **securefile** LOBs use these commands:

     ```
     select sum(bytes)/1024/1024 size_in_mb from dba_segments;
     ALTER TABLE vc_persistent_record MOVE TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_REC_PATH_VER REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_RECORD_DIRECTORY REBUILD TABLESPACE users;
     ALTER INDEX SYS_XXXXXXX* REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSIST_REC_COMMIT_PATH REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_RECORD_VERSION REBUILD TABLESPACE users;
     ALTER TABLE vc_persistent_record MOVE LOB(persistent_data) STORE AS (TABLESPACE users);
     select sum(bytes)/1024/1024 size_in_mb from dba_segments;
     ```
   - For **basicfile** LOBs use these commands:

     ```
     select sum(bytes)/1024/1024 size_in_mb from dba_segments;
     ALTER TABLE vc_persistent_record MOVE TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_REC_PATH_VER REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_RECORD_DIRECTORY REBUILD TABLESPACE users;
     ALTER INDEX SYS_XXXXXXX* REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSIST_REC_COMMIT_PATH REBUILD TABLESPACE users;
     ALTER INDEX VC_PERSISTENT_RECORD_VERSION REBUILD TABLESPACE users;
     ALTER TABLE VC_PERSISTENT_RECORD MODIFY LOB (persistent_data) (SHRINK SPACE);
     select sum(bytes)/1024/1024 size_in_mb from dba_segments;
     ```

   Note: The database generates a SYS_XXXXXXX index file. Check the indexes in the `vc_persistent_record` table for the index name.
6. **Optional:** Free the unused space so that it can be reused. Note that the process might not reduce the size of the underlying data files. To start the process of reclaiming disc storage space (if wanted), a database administrator must complete database maintenance tasks.

```
ALTER TABLE vc_persistent_record ENABLE ROW MOVEMENT;
ALTER TABLE vc_persistent_record SHRINK SPACE;
ALTER TABLE vc_persistent_record DISABLE ROW MOVEMENT;
```

What to do next

1. Verify that the space is reduced:

   ```
   select sum(bytes)/1024/1024 size_in_mb from dba_segments;
   ```

2. Set the **versioned_config.upgrade.enabled** parameter back to false in the `conf/server/installed.properties` file.

Releasing LOB space in MySQL or Maria databases

You can release the large object (LOB) storage in the MySQL or Maria databases.

Before you begin

Consider the following points when you plan an upgrade:

- You might be able to avoid the downtime by running the alter table statement before you upgrade:

  ```
  alter table vc_persistent_record modify persistent_data longtext null, add compessed_data longblob null,
   lock=none
  ```

  If this command fails, an online change of the table before upgrade isn't possible with your version of MySQL. Downtime is required during the upgrade.
- Based on the MySQL version you are using, the upgrade or pre-upgrade table change temporarily uses additional space to rebuild the table.
- For bext results, run the statement on a table clone to estimate the upgrade time.
- Use the alter table command only if the table does not include a compressed_data column.

About this task

Release LOB storage for MySQL and Maria databases using the following steps:

1. Set the **versioned_config.upgrade.enabled=true** property in the `conf/server/installed.properties` file.
2. Restart the HCL DevOps Deploy (Deploy) server.
3. Check on the progress of the conversion:

   ```
   select count(*) from vc_persistent_record where persistent_data is not null
   ```

   The compression process is complete when the row count reported by the query is 0.

4. Check the database size:

   ```
   select table_schema, sum((data_length+index_length)/1024/1024) AS MB from information_schema.tables
    group by 1
   ```

5. Compress the database table.
    - Use the optimize statement for MySQL v5.6 and later, and MariaDB v10.0 and later:

      ```
      optimize table vc_persistent_record
      ```

      See the MySQL documentation about using the optimizing statement: https://dev.mysql.com/doc/refman/5.7/en/optimize-table.html.

    - For all other versions of MySQL, complete these steps:
        a. Backup your database:

           ```
           mysqldump -h hostname -u username -p password --databases dbname > ~/mysql-backup.sql
           ```

        b. Create a new database and restore the old database:

           ```
           mysql -h hostname -u username -p password < mysql-backup.sql
           ```

      Note: The backup and restore must be done when the system is offline to prevent loss of new data while restoring.

What to do next

1. Verify that the space is reduced:

   ```
   select table_schema, sum((data_length+index_length)/1024/1024) AS MB from information_schema.tables
    group by 1
   ```

2. Set the **versioned_config.upgrade.enabled** parameter back to false in the `conf/server/installed.properties` file.

Releasing LOB space in Microsoft SQL Server databases

You can release the large object (LOB) storage in the Microsoft SQL Server database.

Before you begin

Plan downtime for upgrading HCL DevOps Deploy (Deploy) and releasing LOB space.

About this task

Release LOB storage for Microsoft SQL Server database using the following steps:

1. Set the **versioned_config.upgrade.enabled=true** parameter in the `conf/server/installed.properties` file.
2. Restart the Deploy server.
3. Check on the progress of the conversion:

   ```
   select count(*) from vc_persistent_record where persistent_data is not null
   ```

   The compression process is complete when the row count reported by the query is 0.

4. Check the database size:

```
EXEC sp_spaceused @updateusage = N'TRUE';
```

5. Compress the database table:

```
DBCC SHRINKDATABASE (dbname, 0);
DBCC SHRINKDATABASE (dbname);
```

What to do next

1. Verify that the space is reduced:

```
EXEC sp_spaceused @updateusage = N'TRUE';
```

2. Set the **versioned_config.upgrade.enabled** parameter back to false in the `conf/server/installed.properties` file.

Releasing LOB space in DB2 databases

You can release the large object (LOB) storage in the DB2 database.

Before you begin

Plan downtime for upgrading HCL DevOps Deploy (Deploy) and releasing LOB space.

About this task

Release LOB storage for DB2 database using the following steps:

1. Set the **versioned_config.upgrade.enabled=true** parameter in the `conf/server/installed.properties` file.
2. Restart the Deploy server.
3. Check on the progress of the conversion:

```
select count(*) from vc_persistent_record where persistent_data is not null
```

   The compression process is complete when the row count reported by the query is 0.

4. Catalog the Deploy DB2 database:

```
db2 catalog tcpip node dbname remote ipAddress server port
db2 catalog database dbname at node dbname
```

   For example, enter the following command:

```
db2 catalog tcpip node mydb remote 10.134.119.178 server 48408
```

   You must catalog the database only once on the client system. If you have to catalog a different database, either use a different node and database name (mydb in this example) or uncatalog the cataloged database:

```
db2 uncatalog database mydb
db2 uncatalog node mydb
```

5. Connect to the cataloged database:

   ```
   db2 connect to mydb user db2inst1 using db2inst1
   ```

   Catalog the database every time you start up the DB2 command window.
6. Check the database size:

   ```
   db2 -v "CALL GET_DBSIZE_INFO(?, ?, ?, -2)"
   ```

7. Check how much space each tablespace is using:

   ```
   db2 -v "SELECT varchar(tbsp_name, 30) as tbsp_name, decimal(tbsp_free_pages * tbsp_page_size)/1024/1024
    as free_space_mb, decimal(tbsp_total_pages * tbsp_page_size)/1024/1024 as total_space_mb FROM
    TABLE(MON_GET_TABLESPACE('',-2))"
   ```

8. Compress the database table:

   ```
   db2 -v "REORG TABLE VC_PERSISTENT_RECORD LONGLOBDATA"
   db2 -v "ALTER TABLESPACE USERSPACE1 LOWER HIGH WATER MARK"
   db2 -v "ALTER TABLESPACE USERSPACE1 REDUCE MAX"
   ```

   The preceding example includes the assumption that the vc_persistent_record table is in a tablespace called USERSPACE1. Replace USERSPACE1 with the tablespace name that contains the vc_persistent_record table.

   Note: The ALTER TABLESPACE commands must only be performed if the Deploy is offline.

What to do next

1. Verify that the space is reduced:

   ```
   db2 -v "CALL GET_DBSIZE_INFO(?, ?, ?, -2)"
   ```

2. Set the **versioned_config.upgrade.enabled** parameter back to false in the `conf/server/installed.properties` file.

Scanning and removing orphaned Codestation blob files

The csorphanscan tool in the `/bin` directory scans for orphaned Codestation blob files and relocates them to an orphan's directory.

About this task

An orphaned Codestation blob file is a file that contains data for a component version that is no longer referred to by any component version currently recorded in the database. Such files use disk space but can never be accessed by a deployment process.

By default, the tool runs in a dry-run mode such that the file system is not modified and the tool only reports statistics. The tool requires some time to execute.

The csorphanscan tool supports only disk-based blob storage. S3 storage is unsupported.

The csorphanscan tool can run while the server is online. In an HA cluster, it must be run on only one cluster member.

To scan and remove orphaned Codestation blob files:

1. Run the csorphanscan command without the --move-orphans argument to report statistics.

   This run takes some time. In an HA cluster, all members share the same database and Codestation storage, so run the csorphanscan tool on only one cluster member.

2. Run the csorphanscan command with the --move-orphans argument to relocate orphans.
   This process takes some time. The orphan files are relocated to the `<appdata>/var/repository/blob/orphans` folder. In an HA cluster, all members share the same database and Codestation storage, so run the csorphanscan tool on only one cluster member.

   You can also use the following flags with the csorphanscan tool:

Option	Description
`--orphan-list-file <path>`	Write a list of orphaned files to a file in the specified path.
`--workers N`	Set the number of concurrent workers. The default number is 4, and the minimum number is 1. Values greater than 4 might not reduce tool run time.
`--memory N[k|m|g]`	Set the amount of heap memory. The default size is 2 GB. Instances with very large numbers of component versions stored in Codestation might require additional memory. The default value is sufficient for at least 400K component versions.

3. Back up the `<appdata>/var/repository/blob/orphans` file and delete it to free the disk space. You must use the file deletion tools of your operating system. In an HA cluster, all members share the same Codestation storage, so perform this procedure on only one cluster member.

   Note: You must never manually delete any other `<appdata>` content outside the orphan's directory.

Cleaning up targets

If you have storage issues and feel that you need to cleanup some unnecessary stored data, you can use cleanup settings to cleanup your database and file system storage.

You can define the cleanup settings for the deployment history, generic process history, snapshots, and component versions from the **System Settings** page. Before you define the cleanup settings, know how these cleanup settings affect your database and file system storage:

- **Deployment History Cleanup:** This setting primarily deletes database files or entries. However, it also removes log files from the file system.
- **Component Version Cleanup:** This setting mainly cleans up the file system, but it also deletes some associated database files.
- **Generic Process History Cleanup** and **Snapshot Cleanup**: Both of these settings exclusively delete database files.
- **Process Step Log Cleanup**: This setting cleans up the file system exclusively.

Creating a rotating audit log file

You can create rotating audit log file at a regular interval, such as every day.

About this task

You can create a rotating audit log file by adding the `audit-log-to-file` appenders to the log4j.properties. Starting with 7.0.1.1, you can find the appenders in the log4j.properties file. Go to **Settings > Logging**, and uncomment the respective appenders.

The following code fragment shows the `audit-log-to-file` appenders to be used to create a rotating audit log file:

```
log4j.appender.audit-log-to-file=org.apache.log4j.DailyRollingFileAppender
log4j.appender.audit-log-to-file.File=../var/log/audit.log
log4j.appender.audit-log-to-file.MaxFileSize=100MB
log4j.appender.audit-log-to-file.DatePattern='.'yyyy-MM-dd
log4j.appender.audit-log-to-file.layout=org.apache.log4j.PatternLayout
log4j.appender.audit-log-to-file.layout.ConversionPattern=%m%n
log4j.logger.AuditLog=DEBUG, audit-log-to-file
log4j.additivity.AuditLog=false
```

1. Edit the `log4j.properties` file.
   Use a text editor to edit the file. You can also access the file by clicking **Settings > Logging**.
2. Modify the appenders.
3. Restart the server.
4. Verify the logs generating in the `/var/log/audit.log` file.

What to do next

This feature works especially well with audit log cleanup enabled. Go to **Settings > System > Audit Log > Audit Log settings > Enable Audit Log Cleanup** to enable audit log cleanup. Additionally, configure your audit log settings from the System settings on page 2086 as required.

Cleaning up deployment history files

You can clean up the deployment history files to reduce the database storage that HCL DevOps Deploy (Deploy) uses.

About this task

You can enable the deployment history cleanup and specify when the cleanup process begins, the duration of the cleanup, and the number of days to retain the deployment history. You can clean up the deployment history files in the following three ways:

- By configuring the cleanup settings from the system settings. See Configuring the system settings on page 2135.
- By configuring the cleanup settings for specific environments. See Configuring individual environment settings on page 2135
- By configuring the cleanup setting for multiple environments.Configuring bulk cleanup settings on page 2147

Important: The deployment history cleanup process deletes all records in the database and the file system that are related to the deployments that you're deleting. The cleanup cannot be undone. It is important that you carefully choose the items that you delete.

For the cleanup process to start and run according to specifications, the cleanup process must be turned ON in by your administrator in **System Settings** and **Use Deployment History Cleanup** must be selected in environment settings. By default, the deployment history cleanup settings in both the administrator and environment settings are turned off.

The following table shows how the cleanup configuration works based on your selection:

Table 696. Enabling deployment history cleanup

Enable Deployment History Cleanup option in System Settings (administrator's setting)	Use Deployment History Cleanup check box in environment settings	Result
On	Selected	Cleanup starts as specified for deployments that are associated with this environment.
On	Cleared	Cleanup does not start for deployments that are associated for this environment.
Off	Selected	Cleanup does not start for deployments that are associated with this environment.

For example, you have two environments; environment A and environment B and your administrator has enabled deployment history cleanup settings in **System Settings**. In the environment B settings, the **Use Deployment History Cleanup** check box is cleared. The **Use Deployment History Cleanup** check box is selected for environment A. When the cleanup process starts, the deployments history associated with environment A will only be deleted per your configuration.

While the administrator has the **Enable Deployment History Cleanup** option turned off, even when a user selects **Use Deployment History Cleanup** for an environment, the deployment history cleanup process remains turned off.

An example usage of the cleanup configuration fields are explained in Example usage on page 2136.

Note: Despite these settings, deployment history cleanup will not work when:

- The deployments are associated with an active inventory of any type, such as deployed component version, resource, resource configuration, and environment properties.
- The server is in an HA cluster and one of the instances is actively running a deployment history clean up job.

Configuring the system settings

Before you begin

Ensure that you have the following permissions to edit the system settings:

- **Web UI > Settings Tab**
- **Server Configuration > Edit Basic System Settings**

1. In Deploy, click the **Settings** tab, and then in the **System** column, click **System Settings**.
2. Scroll to the **Process History Cleanup** area.
3. Set **Enable Deployment History Cleanup** option to **On**.
4. In **Daily Cleanup Start Time**, select the hour to start the daily cleanup process.
5. In **Daily Cleanup Duration (hours)**, specify the number of hours for the cleanup process to run.
6. In **Days to Retain Deployment History**, specify the default number of days to keep the deployment history before the cleanup process removes the history.

Results

When cleanup begins, a progress indicator shows details about the cleanup process.

Configuring individual environment settings

Before you begin

Ensure that you have the following permissions to edit the deployment history files for individual environments:

- **Environment > Edit > Manage Environment Cleanup**
- **Web UI > Applications Tab**

1. Click **Applications**, click an application, click an environment, click **Configuration**, and then click **Basic Settings**.
2. Select **Use Deployment History Cleanup**.
3. Perform any one of the following steps:
    - Select **Use Default Deployment History Retention** to keep deployment history details for the number of days that the administrator specified.
    - Clear the **Use Default Deployment History Retention** option and specify the number of days to keep deployment history details in the **Days to Retain Deployment History** field.

Results

When cleanup begins, a progress indicator shows details about the cleanup process.

Example usage

About this task

This example helps you understand how cleanup works in Deploy.

If the date is 14 December 2016, and you have used the following settings:

- **Enable Deployment History Cleanup** in System Settings: Enabled
- **Daily Cleanup Start Time** in System Settings: `9:00 PM`
- **Daily Cleanup Duration (hours)** in System Settings: `3`
- **Days to Retain Deployment History** in System Settings: `15`
- **Use Deployment History Cleanup** in the environment settings: Enabled
- **Use Default Deployment History Retention** in the environment settings: Enabled

At 9:00 PM on 14 December 2016, the deployment history begins to run for 3 hours. During the 3 hours, all records of the deployment that are 15 days old and older are deleted. The three hours might be insufficient to delete all records that meet the cleanup criteria. In this case, cleanup resumes operations the next day as scheduled. If you need to clear more disk space, increase the cleanup duration.

Notes:

- If the cleanup settings are changed before 14 December 9:00 PM, then items that are deleted will not be the same as in this example.
- You can make changes to the settings after this date and change it multiple times.
- At any time, the administrator or users with privileges can change whether the cleanup process runs, what cleanup removes, when cleanup starts, and how long cleanup runs.

Tip: It is not recommended to cleanup all history at once. For example, if your goal is to keep certain data for 10 days before it is cleaned up, you should slowly reduce the number of days to keep, so that you do not attempt to cleanup the last 4 years all at the same time.

Cleaning up component versions

You can configure the server to reduce the number of component versions by archiving them. You can specify artifacts cleanup settings from the System Settings, or cleanup settings for individual components, and environments.

About this task

The server archives a full or incremental component version if the version does not meet any of the criteria for versions to keep. When archiving versions, the server retrieves the settings from the **Settings > System Settings > Artifact Cleanup** options first. Then it retrieves the settings for the individual components, which override the system settings. Finally, it checks the settings on the environments to which the component has been deployed. If any of these settings specify to keep the component version, it is not archived.

These settings include a number of days for which to retain versions and a number of versions to keep. The system default component settings, and the component settings refer to the creation date of the version. For example, if the system default settings or component settings keep versions for 30 days, a version is not archived until its creation date is more than 30 days in the past. Similarly, if the system default settings or component settings keep the 10 most recent versions, a version is not archived until there are 10 more recently created versions. Component settings override system default settings for that component.

By contrast, the environment settings refer to the most recent deployment date of the version. For example, if the environment settings keep versions for 30 days, a version is not archived if it has been deployed in the last 30 days. If an environment does not have settings, the server uses the system settings instead. For example, if the server settings keep versions for 30 days and the version was created more than 30 days ago but was deployed to an environment within 30 days, the version is kept.

The server uses the following criteria to determine which versions to keep and which to archive:

First, the server checks the settings on the component. If the component has settings to keep versions, the server uses those settings. Otherwise, the server uses the defaults in the server settings.

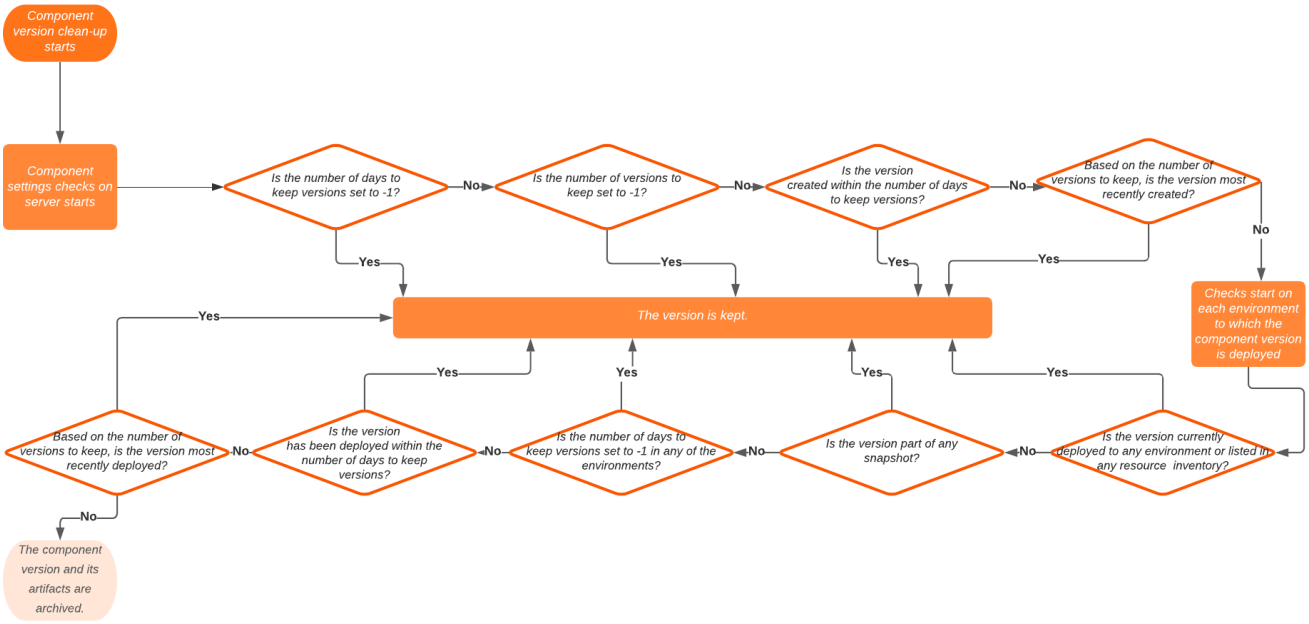
- If the number of days to keep versions is set to -1, the version is kept.
- If the number of versions to keep is set to -1, the version is kept.
- If the version was created within the number of days to keep versions, it is kept.
- Based on the number of versions to keep, if the version is in the most recently created versions, it is kept.

Then, the server checks the settings for each environment to which the component version has been deployed. If an environment does not have settings, the server uses the system settings for that environment. In the context of environments, the settings refer not to the creation date of the component but the date of the most recent deployment of that component. For example, if the system settings keep the 20 most recent versions, and the version is one of the 20 most recently deployed versions in an environment, the version is kept. Using these settings, the server uses the following criteria:

- If the version is currently deployed to any environment or is listed in the inventory of any resource, it is kept.
- If the version is part of any snapshot, it is kept.
- If the number of days to keep versions is set to -1 in any of the environments, the version is kept.
- If the number of versions to keep is set to -1 in any of the environments, the version is kept.
- If the version has been deployed within the number of days to keep versions, it is kept.
- Based on the number of versions to keep, if the version is in the most recently deployed versions, it is kept.

If none of these criteria are true, the component version and the artifacts in it are archived, even if the version was deployed to an environment in the past.

The following diagram is a graphical representation of the checks performed on the server for the clean-up of component versions.

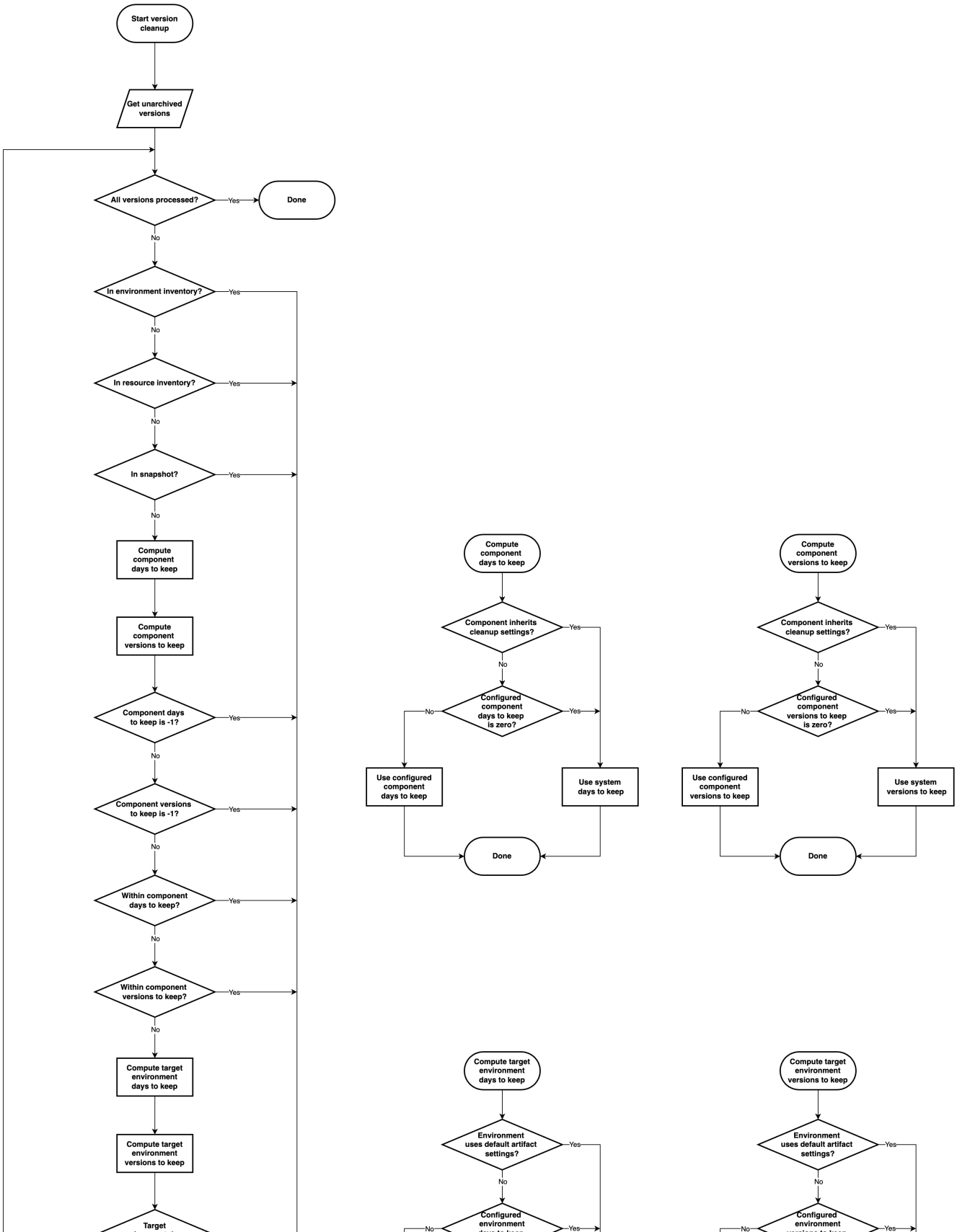

Component version clean-up starts
Component settings checks on server starts
Is the number of days to keep versions set to -1?
No
Is the number of versions to keep set to -1?
No
Is the version created within the number of days to keep versions?
No
Based on the number of versions to keep, is the version most recently created?
No
Yes
Yes
Yes
Yes
Checks start on each environment to which the component version is deployed
The version is kept.
Yes
Yes
Yes
Yes
Yes
Based on the number of versions to keep, is the version most recently deployed?
No
Is the version has been deployed within the number of days to keep versions?
No
Is the number of days to keep versions set to -1 in any of the environments?
No
Is the version part of any snapshot?
No
Is the version currently deployed to any environment or listed in any resource inventory?
No
The component version and its artifacts are archived.

Note:

- The server stores full copies of each file in each component version, even if the files are duplicates.
- When you perform an artifact cleanup, upcoming or scheduled processes are not taken into consideration. You must be aware that the artifact cleanup removes all versions according to the schedule, regardless of the

versions that might be used in upcoming scheduled deployments. Schedule artifact cleanups with upcoming deployments in the timing to avoid deployment problems.

This additional flowchart provides a more comprehensive view of the cleanup, showing the cleanup steps in detail.



1. Specify the artifacts cleanup settings:

    a. Click **Settings > System Settings**.

    b. Under **Artifact Cleanup**, in the **Daily Cleanup Start Time** field, specify the hour of the day to start the cleanup process.

    c. In the **Default Application Snapshot Retention (days)** field, specify the minimum age in days to save snapshots.
    For more information, see Cleaning up snapshots on page 2143.

    d. In the **Default Version Retention for Component (days)** field, specify the minimum age in days to save a component version.
    Example
    For example, to keep all component versions that are 5 days old or newer, specify `5`. If you specify `-1`, all versions are saved, regardless of the setting in the **Default Number of Versions to Retain for Environment** field.

    e. In the **Default Number of Versions to Retain for Component** field, specify the minimum number of versions to keep.
    Example
    For example, to save the five most recent component versions, specify `5`. If you specify `-1`, all versions are saved, regardless of the settings in the **Default Version Retention for Component (days)** field.

    When the server runs the cleanup, it deletes or archives all component versions that are older than the minimum age. However, the server retains at least the number of versions in the **Default Number of Versions to Retain for Component** field.

    f. In the **Default Version Retention for Environment (days)** field, specify the minimum age in days to save a environment version.
    Example
    For example, to keep all environment versions that have been deployed with in the last 5 days, specify `5`. If you specify `-1`, all versions are saved, regardless of the setting in the **Default Number of Versions to Retain for Environment** field.

    g. In the **Default Number of Versions to Retain for Environment** field, specify the minimum number of environment versions to keep.
    Example
    For example, to save the five most recently deployed versions, specify `5`. If you specify `-1`, all versions are saved, regardless of the settings in the **Default Version Retention for Environment (days)** field.

    When the server runs the cleanup, it deletes or archives all versions that have been deployed prior to the minimum age. However, the server retains at least the number of versions in the **Default Number of Versions to Retain for Environment** field.

h. In the **Archive Path** field, specify the path of the location on the server to write compressed versions of each version that is archived.
If you do not specify a path, then the server does not store the component versions that it removes.

i. To preview the component versions to be archived the next time that an archive file is created, on the **System Settings** tab, click the **Preview Version Cleanup** link, and select a component.

j. Click **Save**.

2. **Optional:** Specify cleanup settings for individual components.
These settings override the system settings and are editable only if you are assigned to a role that has the **Manage Component Cleanup** setting enabled.

a. Click **Components**, click a component, go to the **Configuration** tab for the component, and then click **Basic Settings**.

b. Clear the **Inherit Cleanup Settings** check box, and specify the following cleanup settings for the selected components:
   - Number of days you want to keep the versions in the **Days to Keep Versions** field.
   - Number of versions you want to keep in the **Number of Versions to Keep** field.

c. Click **Save**.

3. **Optional:** Specify cleanup settings for individual environments.
These settings override the system settings and are editable only if you are assigned to a role that has the **Manage Environment Cleanup** setting enabled.

a. Click **Applications**, click an application, click an environment, click **Configuration**, and then click **Basic Settings**.

b. Clear the **Use Default Artifact Cleanup Settings** check box and provide values for the following fields:

Field	Description
Days to Retain Application Snapshots	Specify the number of days to retain snapshots after they are not the most recently deployed snapshot in this environment. The most recently deployed snapshot is always retained. To retain application snapshots indefinitely, enter -1. To use the system default retention value, enter 0.
Days to Retain Versions	Specify the minimum age in days to save a component version. If you set both component and environment settings, the higher value is used.

Field	Description
Number of Versions to Retain	Specify the minimum number of versions to keep.

c. Click **Save**.

Results

The artifacts, components, and environments are archived as per the cleanup settings defined by you.

What to do next

You may find environment inventory entries for components that no longer have resources mapped with an environment. These environment inventory entries will cause component versions to be unavailable for cleanup. To remove inventory entries of an environment, open the environment options, and click **Sync Inventory**. A message displays the number of entries removed from the environment.

Related information

Migrating Codestation from filesystem repositories to S3 repositories on page 2120

Archiving version artifacts to Amazon S3 on page 2122

Cleaning up snapshots

You can configure the server to reduce the number of snapshots by deleting old snapshots. You can specify global cleanup settings and cleanup settings for specific environments.

About this task

Snapshot cleanup runs daily, at the same time as component version cleanup. The server deletes snapshots that do not meet any of the criteria for snapshots to keep. The effect is the same as if you delete the snapshot manually. These snapshots are not archived and are not recoverable.

When cleaning up snapshots, the server retrieves the settings from the **Settings > System Settings > Artifact Cleanup** options first. Then it retrieves the settings for individual environments, which override the system settings. If any of these settings specify to keep the snapshot, it is not deleted.

The settings for snapshot cleanup specify a number of days to keep snapshots. This time period is measured from the day that the snapshot is no longer the most recently deployed snapshot in an environment. For example, assume that the **Default Application Snapshot Retention (days)** system setting is set to 30 days. If a snapshot is deployed on day 1 and another snapshot is deployed on day 10, the first snapshot can not be cleaned up until day 40.

The system setting also refers to the number of days since the snapshot was created. If a snapshot was created fewer days ago than the **Default Application Snapshot Retention (days)** system setting, it is kept.

The criteria for keeping or deleting snapshots are as follows:

- If the snapshot is the most recently deployed snapshot in any environment, it is kept.
- In any environment, if the snapshot was the most recently deployed snapshot within the number of days in the system setting or the environment setting, it is kept.
- If the snapshot age in days is less than the system setting, it is kept.

If none of these criteria are true, the snapshot is deleted, even if it was deployed to an environment in the past.

1. Specify the system snapshot cleanup settings:
    a. Click **Settings > System Settings**.
    b. Under **Artifact Cleanup**, in the **Default Application Snapshot Retention (days)** field, specify how long to retain snapshots, in days.
    c. Click **Save**.

   For information about the other settings in this section, see Cleaning up component versions on page 2136.
2. **Optional:** Specify cleanup settings for individual environments.

   These settings override the system settings.
    a. Click **Applications**, click an application, click an environment, click **Configuration**, and then click **Basic Settings**.
    b. Clear the **Use Default Artifact Cleanup Settings** check box.
    c. In the **Days to Retain Application Snapshots** field, specify how long to keep a snapshot after it is not the most recently deployed snapshot in the environment.
    d. Click **Save**.

Cleaning up generic process history files

You can clean up generic process history files that are generated when you run the process manually.

About this task

Generic processes can be run as part of a deployment process, or independent of deployment processes. Cleaning up of these two types of generic processes is handled differently in HCL DevOps Deploy (Deploy). In the traditional cleanup process, all logs associated with the deployment process, including generic process logs, are cleaned up. See Cleaning up deployment history files on page 2133 for more information. However, logs that are generated by running a generic process manually are cleaned up by using the generic process cleanup settings.

Note: This procedure does not impact the generic process history files that are part of a deployment process.

The administrator enables the generic process history cleanup and specifies when the cleanup process begins, the duration of the cleanup, and the number of days to retain the process history. By default, the generic process history cleanup enablement settings are turned off.

Important: The cleanup cannot be undone. Carefully choose the time and duration for which you want to delete the records.

To enable, disable, or configure generic process history cleanup settings, an administrator completes these steps:

1. In Deploy, click the **Settings** tab, and then in the **System** column, click **System Settings**.
2. Scroll to the **Process History Cleanup** area.
3. Set **Enable Generic Process History Cleanup** to **On**.
4. In **Daily Cleanup Start Time**, select the hour to start the daily cleanup process.
5. In **Daily Cleanup Duration (hours)**, specify the number of hours for the cleanup process to run.
6. In **Days to Retain Generic Process History**, specify the number of days to keep the generic process history before the cleanup process removes the history.

Results

When cleanup begins, a progress indicator shows details about the cleanup process.

Cleaning up process step log files

You can clean up process log files that are generated when you run deployment process steps to free up storage space in the file system.

About this task

When you run an independent process step or process step which is part of a deployment process, the log files are generated. The process step logs are part of the deployment history files, but you can clean up the step log files independent of the deployment history cleanup. By cleaning up the step log files separately, you can retain deployment history and information such as workflow traces and inventory information.

You can clean up the process step logs in the following three ways:

- By configuring the cleanup settings from the system settings. See Configuring the system settings on page 2146.
- By configuring cleanup settings for individual environments. See Configuring individual environment settings on page 2147.
- By configuring cleanup settings for multiple environments. See Configuring bulk cleanup settings on page 2147.

Important: The process step logs cleanup process deletes all logs in the file system that are related to the deployments that you're deleting. The cleanup cannot be undone. It is important that you carefully choose the items that you delete.

For the cleanup process to start and run according to specifications, the cleanup process must be turned on by your administrator in **System Settings** and the **Use Process Step Log Cleanup** option is selected in environment settings. By default, the process step log cleanup settings in both the administrator and environment settings are turned off.

The following table shows how the cleanup configuration works based on your selection:

Table 697. Enabling step log cleanup

Enable Process Step Log Cleanup option in System Settings (administrator's setting)	Use Process Step Log Cleanup check box in environment settings	Result
On	Selected	Cleanup starts as specified for deployments that are associated with this environment.
On	Cleared	Cleanup does not start for deployments that are associated with this environment.
Off	Selected	Cleanup does not start for deployments that are associated with this environment.

An example usage of the cleanup configuration fields is as follows:

If the date is 14 November 2024, and you have used the following settings:

- **Enable Process Step Log Cleanup** in System Settings: Enabled
- **Daily Cleanup Start Time** in System Settings: `9:00 PM`
- **Daily Cleanup Duration (hours)** in System Settings: `3`
- **Days to Retain Process Step Logs** in System Settings: `15`
- **Use Process Step Log Cleanup** in the environment settings: Enabled
- **Use Default Step Log Retention** in the environment settings: Enabled

At 9:00 PM on 14 November 2024, the process step log cleanup begins to run for 3 hours. During the 3 hours, the step logs that are 15 days old and older are deleted.

Notes:

- You can make changes to the settings after this date and can change it multiple times.
- This procedure does not impact the process step log files that are part of a running deployment process.

Configuring the system settings

As a system administrator, you can enable and configure the process step clean up settings to delete the old process step log files.

Before you begin

You must have the following permissions to edit the system settings:

- **Web UI > Settings Tab**
- **Server Configuration > Edit Basic System Settings**

1. In Deploy, click the **Settings** tab, and then in the **System** column, click **System Settings**.
2. Scroll to the **Process History Cleanup** area.
3. Set the **Enable Process Step Log Cleanup** option to **On**.
4. In the **Daily Cleanup Start Time** field, select the hour to start the daily cleanup process.
5. In the **Daily Cleanup Duration (hours)** field, specify the number of hours for the cleanup process to run.
6. In the **Days to Retain Process Step Logs** field, specify the number of days to keep the process log files before the cleanup process deletes the logs.
7. Save your changes.

Results

You have configured the process step log files cleanup in the system settings. When cleanup begins, a progress indicator shows details about the cleanup process.

Configuring individual environment settings

As a user, you can configure the process step clean up settings for a specific environment.

Before you begin

- You must have the following permissions to edit the process step log cleanup for individual environments:
  - **Environment > Edit > Manage Environment Cleanup**
  - **Web UI > Applications Tab**
- The administrator has enabled the **Enable Process Step Log Cleanup** system setting.

1. Click **Applications**, click an application, click an environment, click **Configuration**, and then click **Basic Settings**.
2. Select **Use Process Step Log Cleanup**.
3. Perform any one of these steps:
   - Select **Use Default Step Log Retention** to keep step log files for the number of days that the administrator specified.
   - Clear the **Use Default Step Log Retention** check box and specify the number of days to keep step logs in the **Days to Retain Step Logs** field.
4. Save your changes.

Results

You have configured the process step log files cleanup setting for the environment. When the cleanup begins, a progress indicator shows details about the cleanup process.

Configuring bulk cleanup settings

You can delete components and environments in bulk automatically when they reach a certain number of versions or after certain days.

Before you begin

You must have the following permissions:

- **Server Configuration > Bulk Manage Cleanup**
- **Web UI > Settings Tab**

About this task

You can configure the cleanup criteria for component versions and environments in bulk.

1. On the Web UI, open the **Cleanup Configuration** page by clicking **Settings > Cleanup Configuration**.
   The list of all components is displayed.
2. Configure cleanup setting for components in bulk.

   a. Select components by using any of the following ways:
      - Filter the component list by **Name** and **Cleanup Setting**.
      - Select all displayed rows by clicking **Select All**.

   b. Click **Set Cleanup Configuration**.

   c. Clear the **Inherit Cleanup Settings** check box, and specify the following cleanup settings for the selected components:
      - Number of days you want to keep the versions in the **Days to Keep Versions** field.
      - Number of versions you want to keep in the **Number of Versions to Keep** field.

3. Configure cleanup setting for environments in bulk.

   a. Select **Environments** from the drop-down on the left.

   b. Select environments by using any of the following ways:
      - Filter the environment list by **Environment** and **Application** name.
      - Select all displayed rows by clicking **Select All**.

   c. Click **Set Cleanup Configuration**.

   d. Select **Use Default Artifact Cleanup Settings** to use the artifact cleanup settings that are defined in the system settings.
   If you do not want to use the system settings, then deselect the **Use Default Artifact Cleanup Settings** option and specify the following cleanup settings for the selected environments:

Field	Description
Days to Retain Application Snapshots	Specify the number of days to retain snapshots after they are not the most recently deployed snapshot in this environment.

Field	Description
	The most recently deployed snapshot is always retained. To retain application snapshots indefinitely, enter -1. To use the system default retention value, enter 0.
Days to Retain Versions	Specify the minimum age in days to save a component version. If you set both component and environment settings, the higher value is used.
Number of Versions to Retain	Specify the minimum number of versions to keep.

e. Select **Use Deployment History Cleanup** to delete the deployment history records of the selected environments.
You can select the **Use Default Deploy History Retention** option to use the deployment history cleanup settings that are defined in system settings.
If you do not want to use the system settings, then deselect the **Use Default Deploy History Retention** option and specify the number of days you want to keep the deploymemt history records in the **Days to Retain Deployment History** field.

f. Select **Use Process Step Log Cleanup** to delete the process step logs of the selected environments.
You can select the **Use Default Step Log Retention** option to use the process log cleanup settings that are defined in the system settings.

If you do not want to use the system settings, then deselect the **Use Default Step Log Retention** option and specify the number of days you want to keep the process step logs in the **Days to Retain Step Logs** field.

4. Save your changes.

Results

You have configured the cleanup settings for components and environments in bulk.

Managing Security

HCL DevOps Deploy (Deploy) uses a flexible team-based and role-based security model that maps to your organizational structure.

From a high level, the security system for the server consists of an authentication realm, authorization realm, roles, and teams. The authentication realm verifies the identity of the user or system that is trying to log on to the Deploy server. The authorization realm manages user groups. Roles manage permissions. Teams join users with roles and specify which objects the team can access.

Permissions

Deploy uses the term *permission* in a way that is familiar to most readers: each permission controls access to a product area or product function. Most permissions refer to typical activities such as creating, reading, updating, and deleting. Separate

permissions are available for each type of object on the server, including components, applications, and environments. Permissions define what can be done, not who can do it.

Within each security type, you specify the permissions that each role has. For example, administrator or team lead roles have many permissions, while observer roles have limited permissions.

Security types

Security types divide one type of object into categories. For example, you can set one security type for production environments and a separate security type for development environments for the same object type. Then, you can set different permissions for each type of environment. Security types are optional; you can define security types or use the default security type for all objects.

Roles

A *role* is a set of granted permissions. A developer-type role might have permissions for creating applications but not running them in production environments. Alternatively, a deployer-type role might be able to run applications but not create them. You decide the number of roles and their functions. By default, Deploy provides several roles. The administrator role, for example, is granted permissions for all security types. You must create your own roles and specify the permissions for each role.

When you create a role, that role becomes available on each team. Then, you can assign users to that role for a specific team.

Important: A role by itself does not impart its granted permissions to any actual user. Like permissions, roles define what can be done, not who can do them. Roles and their associated permissions are applied to users by teams.

Teams

A *team* is a construct that associates users or groups with roles. When a user is added to a team, that person is assigned to one or more roles. Users cannot be added to a team unless they have a role assignment. Role members are granted all permissions that are defined for the role. Groups can also be added to roles, in which case all group members are granted the permissions that are defined for the role.

Teams secure resources. To secure an environment, for example, assign a team to it. After the environment is secured, only team members who are assigned a role with an appropriate permission can affect the environment.

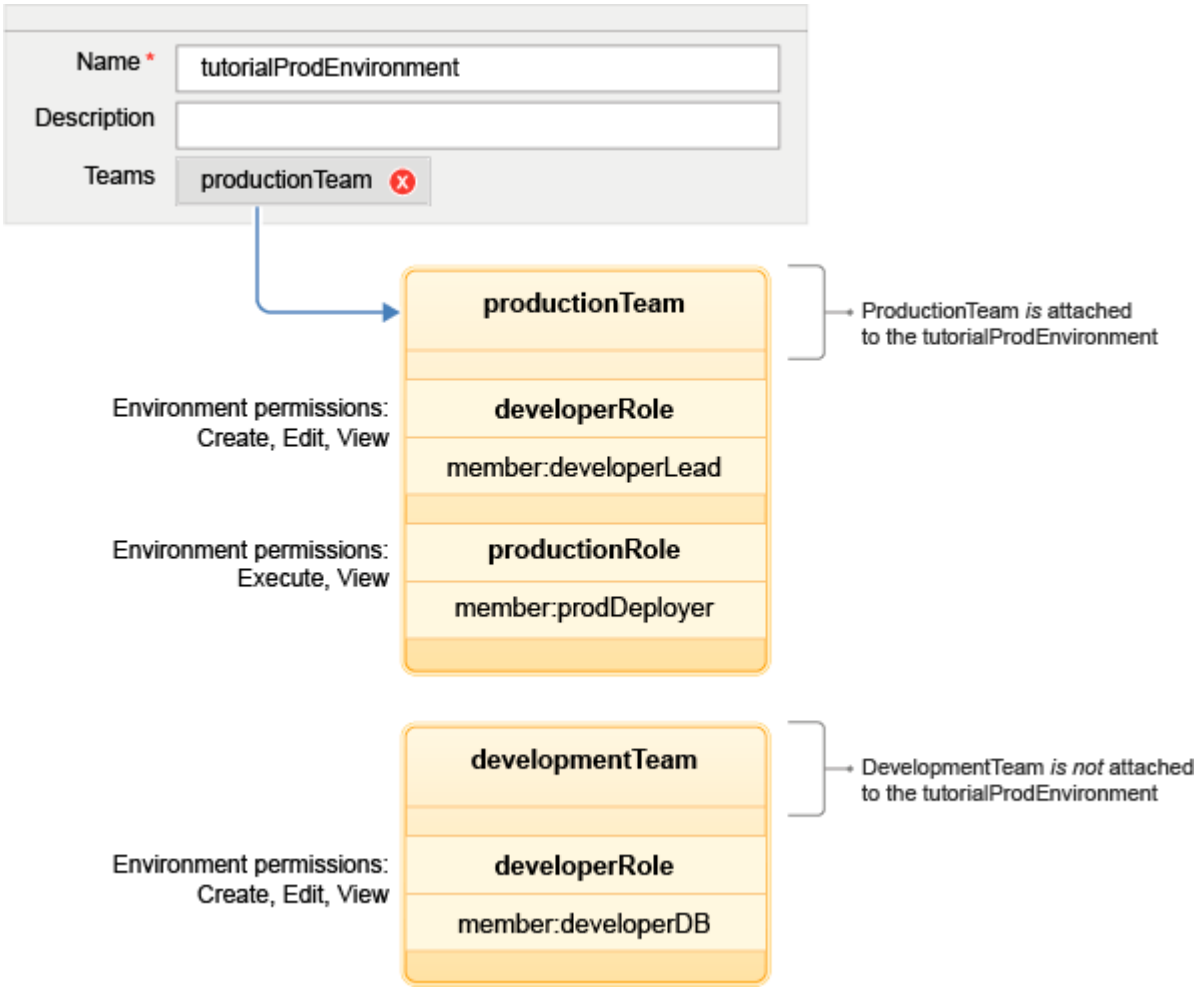
The System Team that comes with Deploy has several users, such as the admin user which is assigned to the Administrator role. The Administrator role has all permissions that are granted for all standard security types and for the server and web UI settings. The System Team cannot be deleted.

The System Team has access to everything on the server according to the permissions in the standard security type. For example, the members of the System Team have the environment permissions that are listed in the Standard Environment security type for all environments.

Resource and object security

To give team users access to a resource or object (such as a component, application, or environment), assign that resource or object to a team. In the following figure, the *productionTeam* team is attached to the *tutorialProdEnvironment* environment. The *productionRole* role has the execute permission for environments. Team members with that role, such as *prodDeployer*, can run applications in the environment. Team members without that role, such as *developerLead*, cannot run applications even though the team is attached to the environment.

Figure 5. Team security permissions



The *developerLead* user can create and modify environments. Additionally, the *developerDB* user can create environments even though the *developmentTeam* team is not attached to this or any other environment.

When a team is assigned to an object, the security type is also determined. By varying security types, you can fine-tune team permissions. For example, a team might have one set of permissions for a component and, by changing the security type, another set for a different component.

Guidelines for setting up server security

Although there is no single best way to set up security for the server, the following steps are sufficient in most cases.

About this task

If you are upgrading from an earlier version, see Migrating security settings.

1. Create an authorization realm for the server. Authorization realms are used by authentication realms to associate users with groups and to determine user access. HCL DevOps Deploy (Deploy) includes an internal database for storing security information and integration with the Lightweight Directory Access Protocol (LDAP). LDAP is a widely used protocol for accessing distributed directory information over IP networks.
2. Create an authentication realm for the server. The authentication realm is used to determine a user's identity within an authorization realm.
3. **Optional:** Create security types.
   Security types classify artifacts such as environments and components by level of security.
4. Create roles and define permissions for them.
5. Create and import users and groups.
6. Create teams, and assign users and groups to them.
7. Assign elements, such as applications and environments, to teams.
   Result
   After you complete the preceding steps, users on the associated teams have access to those elements. The access rights that each user has for those elements depend on the user's role within that team.

SSL configuration

With Secure Sockets Layer (*SSL*) technology, clients and servers can communicate securely by encrypting all communications. Data is encrypted before it is sent and decrypted by the recipient. This communication cannot be deciphered or modified by third-parties. In addition to encryption, SSL can also support authentication.

HCL DevOps Deploy (Deploy) servers and agents communicate via HTTPS. Typically, HTTPS is used for file transfers between the server and an agent. For example, HTTPS is used when an agent is downloading a new plug-in, or when an agent is uploading or downloading version artifacts.

Enabling server identity verification

You can enable extra security to configure the agents to verify the identity of the server for communication that uses the HTTPS protocol.

Before you begin

Upgrade the following plug-ins to the latest versions that are available:

- All source configuration plug-ins
- z/OS® Utility plug-in
- MobileFirst Platform Foundation (formerly Worklight) plug-in
- WebSphere® Application Server - Deployment plug-in

Versions of these plug-ins that were released before March 2016 do not support server identity verification.

Ensure that the keytool utility, which is provided with the Java™ developer kit and is not part of Deploy, is available in the system path.

About this task

When you install the Deploy server, a private key and self-signed certificate with the alias `server` are stored in the *`server_installation_directory`*`/opt/tomcat/conf/tomcat.keystore` file. This certificate is presented to agents, agent relays, and users that connect to the server via HTTPS. Because the certificate that is associated with the private key has a generic distinguished name (DN), you must replace the key so that the agent or agent relay can correctly verify the host name of the server. After you configure the server to present a certificate with a valid host name, you then configure the agents to accept that trusted certificate and to require verification of the host name of the server.

1. Stop the Deploy server.
2. Open a command-line window, and go to the *`server_installation_directory`*`/opt/tomcat/conf` directory.
3. Generate a private key that is associated with the correct host name to use for HTTPS communication. Run a command similar to the following command:

   ```
   keytool -genkeypair -alias serverNewCN -keysize 2048 -sigalg SHA256withRSA -keyalg RSA -keystore
    tomcat.keystore
   ```

   The existing key is stored in the `tomcat.keystore` file with the `server` alias.
4. Edit the `tomcat.key.alias` property in the installed.properties file to specify the alias in the keystore that contains the certificate to use.

   ```
   tomcat.key.alias="serverNewCN"
   ```
5. Edit the following properties in the `secured-installed.properties` file to specify the password for the certificate and keystore to use.

   ```
   server.key.password=<new_certificate_password>
   tomcat.keystore.password=changeit
   ```

   Enter the passwords in plain text, so that it automatically encrypts once the server runs.
6. **Optional:** Create a certificate signing request that uses the new private key, and then use an internal or external certificate authority to sign it.
7. Export the server certificate. Run a command similar to the following command:

   ```
   keytool -exportcert -alias server -keystore tomcat.keystore -file server.cert
   ```
8. Start the Deploy server.
9. If you used a certificate authority that is not already trusted by the agents, copy the server certificate to the agent computer, and then import the certificate into the keystore of the JRE that is used to run the agent process.
   By default, the path to the keystore is `$JAVA_HOME/lib/security/cacerts`. If you use agent relays, repeat this step for all agent relays.
10. If you use agent relays, complete the following steps to configure the agents to verify the identity of the agent relays.

Note: Server identity verification uses the HTTPS, or *codestation*, keystores for the agent relays.

a. Open a command-line window on the agent relay computer, and go to the `/agent_relay_install/conf/` directory.
By default, the *agent-relay-install* location is `/opt/devops-deploy/agentrelay` on Linux™ and `C:\Program Files\devops-deploy\agentrelay` on Windows™.

b. Generate a private key that is associated with the HTTPS connection for the agent relay. Run a command similar to the following command:

```
keytool -genkeypair -alias agentrelay -keysize 2048 -sigalg  SHA256withRSA
    -storepass changeit -keypass changeit -keyalg RSA -keystore codestation.keystore
```

For the `-alias` argument, specify the certificate alias of the *codestation* keystore for the agent relay. The certificate is in the `agentrelay.properties` file for the agent relay, in the property **codestation.cert.alias**. The default value is `agentrelay`.

c. Export the agent relay certificate. Run a command similar to the following command:

```
keytool -exportcert -alias agentrelay -keystore codestation.keystore -file agentrelay.cert
```

d. Copy the agent relay certificate to the agent computer, and then import the certificate into the keystore of the JRE that is used to run the agent process.
By default, the path to the keystore is `$JAVA_HOME/lib/security/cacerts`.

11. Add `verify.server.identity=true` to the *<DEPLOY_AGENT_HOME>*`/conf/agent/installed.properties` file on each agent.
If you use agent relays to cache artifacts, repeat this step for all agent relays.
12. Upgrade each agent or agent relay. If an agent is already upgraded, restart the agent.
Agents can communicate with the server during this process, so you can upgrade agents one at a time.

Supported TLS and SSL protocols and ciphers

HCL DevOps Deploy (Deploy) supports multiple SSL protocols and ciphers for communication among servers.

Deploy supports TLS v1.2, TLS v1.3, and SSL protocols. Deploy supports the SSL v3 protocol only if older agents require it.

Note:

- Limiting and disabling SSL and TLS protocols and ciphers is done at the JVM level rather than the application level.
- Support for TLS v1.0, TLS v1.1, and SSL v2Hello is deprecated.

Deploy uses SSL to communicate between the web UI and server and between the servers that use ActiveMQ. The SSL certificates that control both types of communication use the Java™ KeyStore (JKS) format. These certificates are generated by an RSA key with a 2048-bit length and are signed by a SHA256withRSA algorithm.

By default, the web UI connects on port 8443. The server certificate is in the `opt/tomcat/conf/tomcat.keystore` file. See Configuring SSL on Apache Tomcat and LDAP servers on page 2159 for more information.

The SSL ciphers you can use depend on which ciphers the Java crypto provider of the JRE supports on which you installed your Deploy server, relay and agent. For example, refer to a list of providers by Oracle/OpenJDK Java 11.

The following SSL cipher suites are enabled by default:

Table 698. Default cipher suites

AES_256_GCM_SHA384
AES_128_GCM_SHA256
AES_128_CCM_8_SHA256
AES_128_CCM_SHA256
CHACHA20_POLY1305_SHA256
ECDH_ECDSA_WITH_AES_128_CBC_SHA
ECDH_ECDSA_WITH_AES_128_CBC_SHA256
ECDH_ECDSA_WITH_AES_128_GCM_SHA256
ECDH_ECDSA_WITH_AES_256_CBC_SHA
ECDH_ECDSA_WITH_AES_256_CBC_SHA384
ECDH_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
ECDH_RSA_WITH_AES_128_CBC_SHA
ECDH_RSA_WITH_AES_128_CBC_SHA256
ECDH_RSA_WITH_AES_128_GCM_SHA256
ECDH_RSA_WITH_AES_256_CBC_SHA
ECDH_RSA_WITH_AES_256_CBC_SHA384
ECDH_RSA_WITH_AES_256_GCM_SHA384

Table 698. Default cipher suites (continued)

ECDHE_ECDSA_WITH_AES_128_CBC_SHA
ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE_ECDSA_WITH_AES_256_CBC_SHA
ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
RSA_WITH_AES_128_CBC_SHA
RSA_WITH_AES_128_CBC_SHA256
RSA_WITH_AES_128_GCM_SHA256
RSA_WITH_AES_256_CBC_SHA
RSA_WITH_AES_256_CBC_SHA256
RSA_WITH_AES_256_GCM_SHA384
RSA_WITH_CAMELLIA_128_CBC_SHA
RSA_WITH_CAMELLIA_128_CBC_SHA256
RSA_WITH_CAMELLIA_256_CBC_SHA
RSA_WITH_CAMELLIA_256_CBC_SHA256

Overriding suggested cipher suites

Deploy suggests ciphers based on their availability and what the system identifies as secure. You can use system properties to add specific ciphers or prevent the system to add a cipher automatically to the default cipher list.

To add a cipher to the list of default ciphers, add the following property to the system property:

```
com.urbancode.commons.util.ssl.sslContextEnabledCiphers
```

Example

For example, you can add cipher1 and cipher2 to the default ciphers by adding the following property:

```
com.urbancode.commons.util.ssl.sslContextEnabledCiphers=cipher1,cipher2
```

To prevent the system from adding a cipher to the default cipher list, add the following property:

```
com.urbancode.commons.util.ssl.sslContextIgnoreSuggestedCiphers=true
```

Enforcing the use of a security protocol or set of ciphers

For security reasons, ensure that all SSL connections to and from the HCL DevOps Deploy (Deploy) server uses the TLSv1.2 protocol. To configure SSL globally, follow these instructions under jdk.tls.disabledAlgorithms here.

The Deploy server uses the following communication protocols that rely on SSL at the transport level:

- HTTPS for displaying the user interface
- LDAP to connect to LDAP and Active Directory servers
- JDBC to connect to the database containing the schema of the Deploy server application

Note: Support for TLSv1.0, TLSv1.1, and SSLv2Hello is deprecated.

The following sections explain how to enforce the usage of TLSv1.2.

Environment

Enforcing TLSv1.2 is possible if the Deploy server, relay, and agent are running a JVM version that supports this protocol. For more information on such JVM versions, see Oracle JRE and JDK Cryptographic Roadmap. Before you enforce TLSv1.2 as the only supported protocol in Deploy, make sure that the external tools like LDAP servers, database servers, and middleware that the plug-ins deploy to, also support TLSv1.2 or else the connections will fail.

Deploy server: HTTPS protocol

The HTTPS protocol is configured in the `devops-deploy/server/opt/tomcat/conf/server.xml` file. The `server.xml` file contains the following relevant entries:

```
    <Connector port="${install.server.web.https.port}"
               address="${install.server.web.ip}"
               server="SERVER"
               maxThreads="150"
               enableLookups="false"
               acceptCount="100"
               debug="0"
               connectionTimeout="20000"
               disableUploadTimeout="true"
               compression="1024"
               noCompressionUserAgents="gozilla, traviata"
               compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain,application/json"
               algorithm="${install.server.ssl.algorithm}"
               SSLEnabled="true"
               scheme="https"
               secure="true"
               clientAuth="false"
```

```
               URIEncoding="UTF-8"
               ciphers="${install.server.ssl.enabledCiphers}"
               sslEnabledProtocols="${install.server.ssl.enabledProtocols}"
               keystoreFile="${tomcat.keystore.file}"
               keyAlias="${tomcat.key.alias}"
               keystorePass="${tomcat.keystore.password}" />
```

To enforce TLSv1.2, add or modify the `install.server.ssl.enabledProtocols=TLSv1.2` property in the `devops-deploy/server/conf/server/installed.properties` file. Note that this property might be missing, or it might be present and have a different value, including an empty value. The property is referred to as: `${install.server.ssl.enabledProtocols}` in the `server.xml` and `installed.properties` files.

Deploy Server: LDAP integration

The Deploy server uses the LDAP client that is contained in the JRE that is running on the server. You must specify a URL for the LDAP server, which uses the `ldaps://` protocol that uses port 636 but default to connect through SSL or TLS.

Note: The `install.server.ssl.enabledProtocols` parameter changes the SSLContextParameters which might affect the LDAP connection that is opened.

Deploy Server: JDBC connection to the database server

JDBC is used to connect Deploy to the backend database.

DB2

Refer to the DB2 documentation - Configuring SSL support in a DB2 instance for detailed information. Configure the DB2 instance according to the instructions, and test after you compete the configuration.

Postgres

Connection string: `connectionString=jdbc:postgresql://dbhost/dummy?ssl=true jdbcDriver=org.postgresql.Driver jdbcJar=C:\devops-deploy\agent\postgresql-42.1.4.jar`.

Import the database certificate to the Deploy Agent Java keystore. The latest available plug-in is version 13, see SQL JDBC plug-in for Deploy.

Microsoft SQL Server

1. Ensure that your Microsoft SQL server supports TLSv1.2 and have the right FixLevel and KB 3052404 installed on it.
2. Enable TLSv1.2.
    a. Depending on the Microsoft SQL Server JDBC type 4 driver version, you can pass in the JDBC a connection string that enforces TLSv1.2, that is sslProtocol=TLSv1.2 protocol.
    b. You can enforce TLSv1.2 at the JRE level that runs the Deploy server. Enabling TLSv1.2 has the following results:
        - Outgoing connections to the LDAP database all are made with TLSv1.2.
        - Incoming connections from Deploy agents, Deploy agent relays, and WebUI require TLS v1.2.

These results can be undesirable if one of the involved components does not support TLSv1.2. You must make sure that the JDBC drivers support TLSv1.2. To ensure this support, change the `jdk.tls.disabledAlgorithms` line in the `jre/lib/security/java.security` file to `jdk.tls.disabledAlgorithms=SSLv3, TLSv1, TLSv1.1`. You can also pass the `devops-deploy/server/bin/set_env JAVA_OPTS` string through a java.security file.

Note: You cannot disable the SSLv2Hello, because it is used as the initial handshake from the Deploy agents with the Deploy server.

c. For the JDBC connection string, refer to the SQL Server 2017 documentation - Understanding SSL Support. The `encrypt=true` property uses TLS/SSL.

Do not blanket `trustServerCertificate=false`. Every SSL certificate from Microsoft SQL server uses the default JRE's trust store `jre/lib/security/cacerts`. Use `hostNameInCertificate=yes` to verify that DNS and `SubjectAltName` in the SSL certificate are the same. For `trustStore=/my/own/trustStore.jks`, do not use the default JRE's trust store `jre/lib/security/cacert`, use a separate one. The `trustStorePassword=changeit` is the password for the trust store.

Oracle

To enforce TLSv1.2 for Oracle12c:

1. Ensure that you have a JDBC driver version higher than 12.2.0.1.
2. In the file `devops-deploy/server/bin/set_env.sh` append the following -D argument `-Doracle.net.ssl_version='1.2'` in the value you currently set for JAVA_OPTS variable.
3. Restart the Deploy server.

Deploy Agent

To add TLSv1.2 protocol to Deploy agents using HTTPS communication with the Deploy server, add `-Djdk.tls.client.protocols=TLSv1.2` line to the `devops-deploy/agent/bin/worker-args.conf` file.

For a step running on Deploy agent, add `-Djdk.tls.client.protocols=TLSv1.2` line to the `devops-deploy/agent/conf/plugin-javaopts.conf` file.

Configuring SSL on Apache Tomcat and LDAP servers

The steps for configuring secure HTTPS connections with the HCL DevOps Deploy (Deploy) server are similar to the steps for any Java™ Platform, Enterprise Edition server.

Before you begin

To set up SSL security, you must have a certificate for the server. If you use certificates that are self-signed or certificates that are issued by a certificate authority that is not trusted, you must import these certificates into the trust store. A trust store contains trusted certificates from servers and certificate authorities. Import the self-signed certificates, certificate authority certificates, and intermediate certificate authority certificates to the *JRE_install*`/jre/lib/security/`

`cacerts` file. For an example of importing a certificate to a trust store, see the Importing the Certificate Reply from the CA topic in the IBM® SDK, Java™ Technology Edition help.

In the case of LDAP servers, SSL certificates must have valid chains of authority. If you use your own certificate authority, add that certificate authority to the local trust store.

About this task

Because the Deploy server runs on Apache Tomcat, you can refer to the instructions for configuring security on Tomcat on the Apache Tomcat website. You can enable SSL security for Apache Tomcat when you install Deploy. If you enable SSL security during installation, a self-signed certificate is generated and added to the `tomcat.keystore` file. The common name (CN) in the self-signed certificate is set to 0.0.0.0.

Note: The presence of a self-signed certificate is flagged as a warning by many web browsers when you connect to a Deploy server that uses a self-signed certificate in the `tomcat.keystore` file.

Setting up SSL security for the server involves these general steps:

1. Transfer the files for the certificate to the computer that hosts the Deploy server.
2. Add the certificate to the server keystore.
   The server has a default keystore in the *`server_install`*`/opt/tomcat/conf/tomcat.keystore` file. The default password for this keystore is `changeit`.
3. Edit the `tomcat.key.alias` property in the `installed.properties` file.
   ```
   tomcat.key.alias=alias_of_your_certificate_in_tomcat_keystore
   ```
4. Restart the server.
5. Similarly, add the same certificate to the keystore of each agent and agent relay.
   Example
   For example, the default location of an agent keystore is *`agent_install`*`/conf/agent.keystore`.
6. **Optional:** To configure secure communication between the Deploy server and an LDAP server, add the LDAP server certificate to the *`JRE_install`*`/jre/lib/security/cacerts` file. This file is on the Deploy server. Use the installation folder of the JRE for *`JRE_install`*.

Implementing custom trust stores

A trust store defines the roots of the certificate trust chain. Typically, these are the certificate authority root certificates that sign other certificates. It can also be end entity certificates that are directly trusted. Java® includes a default trust store that contains certificate authority root certificates for many well-known authorities. This trust store is a file called `cacerts` and is contained in the Java installation. The file has the same format as a keystore, but it never contains a private key. It is possible to modify or replace this file to alter the trust roots with the keytool program. Otherwise, a custom trust store must be used instead.

A custom trust store is simply another keystore file containing the custom trust roots. It is even possible to copy `cacerts` and modify the copy to expand the set of trust roots. Unlike with the default trust store, a configuration step is required enable use of a custom trust store.

You can turn on custom trust stores selectively. For example, you can configure agents to always verify the server without the server doing any verification, or you can have the servers verify each other only in the client role.

Note: Trust stores can be used with agents and servers, but not agent relays. The agent relay has no custom trust store support for web agent communication because it does not participate in authenticating communications.

Agent

For web agent communication, the agent permits a custom trust store for validating connections to the server. For the custom trust store to take effect, **verify.server.identity** must be set to **true**. This parameter can be configured in the installer (Agent installation properties), although it can also be set without configuring a custom trust store.

By default, setting this property will use the default trust store in the `cacerts` file. It also performs an extra hostname verification on the server certificate, which is the same as in a web browser.

A custom trust store is configured by setting **agentcomm.truststore** to the path of trust store file and setting **agentcomm.truststore.password** to the keystore file password. If unset, it will default to **changeit**.

```
installed.properties:
  verify.server.identity=true
  agentcomm.truststore=<path>
  agentcomm.truststore.password=<password>
```

Server

The server's use of a trust store is similar to the agent, but has potentially two custom trust stores, depending on the role of the server in the connection. The trust stores are independent and can be selectively enabled, or both can be configured to share the same trust store file.

Server: Server Role

When the server is in the server role, it will authenticate clients, which are both agents and other servers in the same cluster. In this scenario, **verify.client.identity** must be set to **true**.

The default cacerts trust store is used. There is no hostname verification step because client certificates do not contain a hostname. **agentcomm.serverTruststore** is set to the path of the trust store file. **agentcomm.serverTruststore.password** is set to the trust store file password (**changeit** by default). Unlike the agent, the password property is stored in **secured-installed.properties** with other passwords.

Note: Although the trust store is configured for the server role, that server verifies the identity of clients. This explains the asymmetry in the names of this group of properties. When the server is in the client role, the naming is reversed.

```
installed.properties:
  verify.client.identity=true
  agentcomm.serverTruststore=<path>

secured-installed.properties
  agentcomm.serverTruststore.password=<password>
```

Server: Client Role

When the server is the client role, it is only verifying the identity of the other server members of its cluster. In this scenario, **verify.server.identity** must be set to **true**.

The default cacerts trust store is used. As required by the client role, a hostname verification step is also performed. **agentcomm.serverTruststore** is set to the path of the trust store file. **agentcomm.serverTruststore.password** is set to the trust store file password (**changeit** by default).

```
installed.properties:
  verify.server.identity=true
  agentcomm.clientTruststore=<path>

secured-installed.properties
  agentcomm.clientTruststore.password=<password>
```

Upgrading encryption key strength

When you upgrade the server, you can upgrade the SSL encryption.

About this task

This upgrade is optional, but it strengthens server security. The `keytool` application is included in the Java™ developer kit and is not part of HCL DevOps Deploy (Deploy).

1. Upgrade the HTTPS certificate for the server:

    a. On the server, open a command-line window to the *`server_install`*`/opt/tomcat/conf` folder, where *server_install* is the server installation folder.

    b. Delete the existing certificate with the following command:

    ```
    keytool -delete
      -alias tomcat
      -keystore tomcat.keystore
      -storepass changeit
    ```

    Note: Each command in this example must be written on one line. Also, the commands in the following examples must be written on one line. The commands are written on multiple lines for clarity.

    c. Create a certificate with the following command:

    ```
    keytool -genkey
      -alias tomcat
      -storepass changeit
      -keypass changeit
      -keyalg RSA
      -sigalg SHA256withRSA
      -keysize 2048
      -validity 7305
      -dname CN=host_name
      -keystore tomcat.keystore
    ```

For *host_name*, use the host name of the server.

d. Go to the folder *app_data*/conf.

e. Delete the existing entry to the keystore with the following command:

```
keytool -delete
  -alias server
  -keystore server.keystore
  -storepass changeit
```

f. Create an entry to the keystore with the following command:

```
keytool -genkey
  -alias server
  -storepass changeit
  -keypass changeit
  -keyalg RSA
  -sigalg SHA256withRSA
  -keysize 2048
  -validity 7305
  -dname CN=host_name
-keystore server.keystore
```

For *host_name*, use the host name of the server.

2. For each of your agent relays, upgrade the encryption with the following steps:

a. In a command-line window, go to the *relay_install*/conf/jms-relay folder, where *relay_install* is the relay installation folder.

b. Delete the existing entry in the keystore for the agent relay with the following command:

```
keytool -delete
  -alias agentrelay
  -keystore agentrelay.keystore
  -storepass changeit
```

c. Add an entry with the new encryption level with the following command:

```
keytool -genkey
  -alias agentrelay
  -keystore agentrelay.keystore
  -storepass changeit
  -keypass changeit
  -keyalg RSA
  -sigalg SHA256withRSA
  -keysize 2048
  -validity 7305
  -dname CN=host_name
```

For *host_name*, use the host name of the server.

3. If you are using mutual encryption between servers and agents, upgrade the encryption on each agent:

a. In a command-line window, go to the *agent_install*`/conf/jms-relay` folder, where *agent_install* is the agent installation folder.

b. Delete the existing entry in the keystore for the agent with the following command:

```
keytool -delete
  -alias alias
  -keystore keystore_file
  -storepass changeit
```

Replace the *alias* variable with the alias of the server certificate in the agent keystore. The *keystore_file* variable represents the keystore file. The default name of the keystore file is `agent.keystore`. To find the alias, run the following command, and look for the alias name in the output:

```
keytool -list -v -keystore keystore_file
```

Note:

Although agents and relays that use the JMS communication protocol cannot establish connections with HCL DevOps Deploy (Deploy) v7.2.0 and later, the references to JMS persist in some file names, property names, folder names, and directory names.

For example, the following names that are associated with JMS are still used in **Deploy** v7.2 and later: agentrelay.jms_proxy.name, agentrelay.jms_proxy.id, jms-relay.log directory, and jms-relay folder.

Keystores configuration

Server, agents, and agent relays keystores are used for a secured web communication between servers and agents.

Web agents come with and use Public Key Pinning for strong security. Agents and servers identify each other with SSL certificates. The public keys in the certificates are pinned on the initial connection. When a public key is pinned, any certificate that asserts the same identity must be generated with the same public and private key pair that was used to create the original certificate. Connections from clients or servers that do not correctly identify themselves are closed.

Deploy supports pinning multiple server public keys on an agent. Server public key pin information and limit is available in the agent's metadata. To use a new key, generate a key, and deploy the key to the agent. See Add public key pin for a server host to an agent for more information. Atleast one key must be deployed to an agent.

Server keystores

Server keystore locations are defined in the `installed.properties` file.

Server keystore	Description
`agentcomm.keystore`	The keystore contains the certificate and public and private key pair that the server uses to identify itself as the client. Keys are stored in the *serverInstallDir*`/conf/server/s2s-client-identity.keystore` file. These keys are not shared in high-availability environments. The server uses its web

Server keystore	Description
	certificate to identify itself as the server in a connection. The subject DN must be in this format: `CN=ucd-server-`*`EndpointID`*.
`encryption.keystore`	Server property encryption secret keys are stored in the *`serverInstallDir`*`/appdata/conf/encryption.keystore` file. These keys are shared in high-availability environments.
`server.keystore`	The `server.keystore` property points to the server keystore that contains the certificate and public and private key pair that the server uses to identify itself as the server. Keys are stored in the *`serverInstallDir`*`/appdata/conf/server.keystore` file. These keys are shared in high-availability environments.
`tomcat.keystore`	The keystore points to server UI/API HTTP server certificate and key. Keys are stored in the *`serverInstallDir`*`/opt/tomcat/conf/tomcat.keystore` file. These keys are not shared in high-availability environments.
`extra-cacerts.keystore`	The keystore points to the extra TLS/SSL certificates that are separate from the Java cacerts file. By using the extra keystore, you can eliminate the risk of losing custom certificates during Java upgrades. Keys are stored in the *`serverInstallDir`*`/appdata/conf/extra-cacerts.keystore` file. These keys are shared in high-availability environments.

Important: Do not lose the server public/private key. Recovery requires manually clearing the server certificate pins for every web agent. To help prevent loss, make a secure backup of all server keystores.

To override any of the server keystore passwords, see Changing passwords for the server KeyStore on page 2075.

Web agent relay keystores

Each agent relay contains two keystores. The **agentrelay** keystore controls the web connection between the agent relay and the server and agents. The **codestation** keystore controls HTTPS connections to the agent relay and is used with server identity verification. See Enabling server identity verification on page 2152.

The **agentrelay** keystore stores web agent communication client certificate and keys (alias `ucd-agentrelay-id`). The subject DN must be in this format: `CN=ucd-agentRelay-`*`EndpointID`*.

Note: For more information about changing relay keystore password, see Changing passwords for the agent relay keystores on page 2110

Related information

Upgrading encryption key strength on page 2162

Agent metadata on page 2340

Certificate Expiration Enforcement and Auto-Rotation

You can configure the server and agents to mitigate the security risks associated with vulnerable agent certificate expiration dates.

When you install an agent, a self-signed certificate and a key pair are generated with a validity of 20 years. However, you can allow agents to automatically rotate their certificates at regular intervals of time. You can also configure the Deploy server to enforce the certificate expiration of all clients and set their validity.

Automatic rotation of agents certificate

You can configure the agents to generate new certificates periodically by enabling agent certificate auto-rotation.

You can configure agents to generate a new key pair and certificate automatically by using any of the following methods:

- The bulkSetAgentKeypairPolicy on page 2625 udclient command for generating the key pair of agents in bulk.
- The setAgentKeypairPolicy on page 2877 udclient command for generating the key pair of a specific agent. Alternatively, you can add the following parameter to the agent's `installed.properties` file:

  ```
  agentcomm.keypair.maxAgeDays=90
  ```

  The `agentcomm.keypair.maxAgeDays` property defines the validity of the certificate in days. The minimum allowed value is 90 days.

  Note: You must restart the agent if you are editing the agent's `installed.properties` file.

After you define the agents' maxAgeDays property, the agents are configured to automatically rotate their certificates. The agents will attempt to rotate their certificate 30 days before they expire. For example, if you have set the agents' maxAgeDays property as 90, the agents will attempt to rotate their key pair after 60 days.

Important: The unupgraded agents and agents that are often offline intentionally do not rotate certificates or might not rotate certificates promptly. You must ensure that you upgrade and start such agents once before their expiry to rotate their certificates automatically.

Notes:

- The certificate rotation does not interfere with any running deployment process.
- You can verify that the agents have rotated the certificate as configured from the agent's **Agent Security** page.

Enforcing certificate expiration and setting certificate validity

You must enforce the expiry of certificates and set their validity so that the server fails to communicate with the agents with expired certificates and the agents auto-rotate their certificates at a regular interval of time.

Before you begin

You must have completed the following tasks:

- Enabled the automatic rotation of agent certificates. See Automatic rotation of agents certificate on page 2166.
- You must rotate the server and relays client certificates manually before you enable the certificate expiration.

  When you set the enforcement of certificate expiration, the enforcement is set for all client certificates, including the servers' client certificate and the relays' client certificates. However, server and relays cannot rotate their certificates. Therefore, before enabling enforcement you must plan to perform regular rotation of server and relay client certificates manually before they expire.

1. Force the server to reject an expired certificate by adding the following property to the server's `installed.properties` file:

   ```
   agentcomm.clientCertificate.enforceExpiration=true
   ```

   The `agentcomm.clientCertificate.enforceExpiration` property causes the server to enforce the expiration date specified in the agent certificates regardless of their duration. For example, if the agent certificate is valid for 7300 days (20 years), the server accepts it for 7300 days (20 years).

   Notes:
   - If the enforceExpiration property is set to `false` or not set, then the agent's certificate expiration is ignored.
   - If the **verify.client.identity** parameter is `true` in the server's `installed.properties` file, then your certificate authority (CA) TLS rules will be applied. For example, the certificate is signed by the CA, the expiration date is enforced, and so on. These rules are enforced by Java before the enforceExpiration setting gets a chance to run. So, the enforceExpiration setting is redundant with the TLS system. Therefore, you should use `verify.client.identity=true` or `agentcomm.clientCertificate.enforceExpiration=true` setting, but not both.

2. Force the server to limit an acceptable certificate date range for valid certificates by setting the following parameter in the server's `installed.properties` file:

   ```
   agentcomm.clientCertificate.maxAgeDays=730
   ```

   The `agentcomm.clientCertificate.maxAgeDays` property defines the validity of the certificate in days. The minimum allowed value is 90 days.

   Note: Offline agents cannot rotate a certificate. If you have agents that are offline for extended periods of time, do not set `agentcomm.keypair.maxAgeDays` or `agentcomm.clientCertificate.maxAgeDays` to a combination that results in expired or invalid certificates. When these agents are started, they cannot rotate to a valid certificate and connect with the server.

   The following scenarios are explained that describe the server behavior on how you set the agent's and server's maxAgeDays property:

Scenario 1: When you set the agent's maxAgeDays property to 7300 days (20 years) and set the server's maxAgeDays property to 730 days (2 years).

In this scenario, the agent will automatically attempt to rotate the certificate every 20 years. If the agent is offline for 731 days or more, the server will reject the agent's certificate and will fail to communicate with the agent. You must start the agent and manually rotate the agent's certificate.

Scenario 2: When you set the agent's maxAgeDays property to 90 days (3 months) and set the server's maxAgeDays property to 730 days (2 years).

In this scenario, the lower bound is set by the agent. The server accepts the agent's certificate validity of 90 days and the agent will attempt to rotate the certificate every 60 days, which is 30 days before its expiry. If the agent is offline for 91 days, the agent will not be able to rotate its certificate even if the server's maxAgeDays property is set to 730 days. You must start the agent and manually rotate the agent's certificate.

Scenario 3: When you set the agent's maxAgeDays property to 90 days (3 months), have not enabled the enforcement, and the agent is offline for 91 days.

In this scenario, the server will be able to communicate with the agent when the agent is started later. The agent will rotate its certificate when it comes back online. No manual intervention is required in this scenario.

3. Restart the server to apply the changes.
   Result
   The expired agent certificates are now rejected and any existing certificate expire in the time period defined. As in the example used, the certificates will expire in 730 days even if they are not expired according to their stated expiration date.

Results

You have enforced certificate expiration and set the validity of the certificate.

Encryption key tools

Using the encryption key tools, you can manage your encryption keys.

HCL DevOps Deploy (Deploy) includes encryption tools that you can use to create, delete, and rotate your encryption keys. The keycreate, keydelete, and keyrotate tools are available in the `/bin` directory.

Creating an encryption key

Use the **`keycreate`** tool to create a key for secure property encryption without removing previous keys or changing the primary encryption key.

About this task

If you complete the processs, the tool provides instructions to make the new key primary. A new non-primary key cannot be used for property encryption. It must be made primary first.

To create an encryption key:

1. Run the `keycreate` command to create an encryption key.
   It also prints the alias of the new key. You can run the keycreate tool while the server is online. In an HA cluster, it must be run on only one cluster member because all clusters share the same keystore. The usage is keycreate and requires no arguments.
2. Configure the server to use the new key as its primary key.

   Edit the `installed.properties` file and set the **`encryption.keystore.alias`** property to the alias that the `keycreate` command prints. In an HA cluster, each member has its own `installed.properties` file, so you must edit each cluster individually.
3. Restart the server.

   The server loads keys and the primary key setting only at startup. In an HA cluster, you must restart each member.

Results

When this process is complete, the server (or all servers in an HA cluster) uses the new primary key to encrypt new data exclusively. Previous keys can be used only to decrypt previous data.

Rotating an encryption key

The **`keyrotate`** tool uses the primary encryption key to re-encrypt database secure data that was encrypted with other keys.

About this task

You can run the keyrotate tool in dry-run mode such that the database is not modified, and the tool only demonstrates the changes. The --change-database argument is required if you want to carry out the changes in the database. You must take a backup of the database before you use the --change-database argument.

The keyrotate tool can be run while the server is online. In an HA cluster, all members share the same database, so you must run the keyrotate tool on only one cluster member.

The keytool cannot detect if all members of an HA cluster are using the same primary key. If some servers are using different keys, the tool cannot guarantee that all data has been re-encrypted with the current primary key. To resolve this, update the misconfigured cluster members and re-run the tool by using the following procedure:

1. Edit the `installed.properties` file and set the **`encryption.keystore.alias`** property to the correct alias.

   In an HA cluster, each member has its own `installed.properties` file, so each must be edited individually.
2. Restart the updated server.

   The server loads keys and the primary key setting only at startup. In an HA cluster, you must restart each updated member.

Runtime

Generally, the keyrotate tool in dry-run mode requires a significant amount of time to run. Running the tool with --change-database argument might take much longer than the dry-run mode as writes are typically slower than reads. It's important to note that the run times involved with rotating and deleting the keys depends on the resources available in the database, and the amount of data that needs to be processed.

Rotating 3DES keys

When rotating encryption keys, the keyrotate tool displays below error when a 3DES key is found:

```
Cannot upgrade the server: a 3DES property encryption key was detected. 3DES keys are no longer supported. Before you can upgrade the server, you must first upgrade to a 7.2.3 release and use the 'keyrotate' tool to replace references to 3DES keys in your database. Then remove those keys from your property encryption keystore with 'keydelete' or with your JVM 'keytool' utility.
```

To rotate a secure property encryption key:

1. Create and configure a new primary encryption key. See Creating an encryption key on page 2168 for the procedure.
2. Dry-run the keyrotate command and verify that the encryption key is rotating.
   You can use the following supported arguments with the keyrotate command:

Supported arguments	Description
`--workers N`	Sets the number of concurrent workers. The default value is `2`, and the minimum required value is `1`. The database must accept N+1 concurrent connections. Values greater than `8` might not reduce tool execution time.
`--status-interval-secs N`	Prints the status of keyrotation every `N` seconds. The default is `5` seconds. You can set the interval to `0` to disable printing. Changes are logged into the `stdout` file. To troubleshoot errors, check the `stderr` log file.
`--batch-sizes`	Reduces the memory requirements for tables with large rows , such as `wf_workflow_trace` and `vc_persistent_record` tables. For example, you can use the argument as follows: `--batch-sizes "vc_persistent_record=15,wf_workflow_trace=10"` The argument value is a comma-separated list of table names with their reduced batch size. The default batch size for all tables is 50. **Note:** You may need to quote the argument depending on your shell. The quotes used in the example command are not part of the syntax.

Result

The output demonstrates the changes without making actual changes to the database.

3. For a real-run execution, run keyrotate with the --change-database argument to rotate the encryption keys.
   Running the keyrotate tool with the `--change-database` flag enables modification of the database.

   ```
   ./keyrotate --product-home "/opt/devops-deploy/server" --change-database
   ```

   Note: Use the same arguments for both the steps to ensure that the dry-run is as close as the real run.

   Result

   The keyrotate tool lists the keys in keystore, the keynames, and their rotation status. An example output is shown below:

   ```
   key=[desedekey<key1>] refs=0 replaced=0
   key=[aes128key<key2>] refs=2 replaced=2
   key=[aes128key<key3>] refs=0 replaced=0
   key=[aes128key<key4>] refs=12953555 replaced=0
   key=[desedekey<key5>] refs=0 replaced=0
   unknown=[desedekey<key6> refs=10
   ```

   Note: For best results, do not delete previous keys. You might require them to decrypt older data that was encrypted with the previous keys. See Deleting an encryption key on page 2171 for the procedure and an explanation for why this is not recommended. However, you must delete any 3DES keys to be able to use 7.3.0 or newer versions. The keyname in output can be used to delete the 3DES keys.

Deleting an encryption key

The **`keydelete`** command removes a key for secure property encryption.

About this task

You can run the tool in dry-run mode such that the keystore is not modified and the tool only demonstrates the change. The --change-keystore argument is required to cause the tool to complete the change.

The keydelete tool can be run while the server is online. In an HA cluster, you must run the tool on only one cluster member because all members share the same keystore.

Note:

- The keydelete tool cannot remove the primary encryption key.
- The tool can only determine the primary key for the local server member in an HA cluster. It cannot detect if other members have configured a different primary key. For best results, set up the cluster so that all members use the same primary key.

Note: Do not delete keys unless you have a compelling reason to do so. Only the primary key is used for encrypting new secure data. After the previous data is re-encrypted with the new key by using the keyrotate tool, previous keys are not used in normal operation. In some scenarios, you might require previous keys, such as decrypting data

that you restore from a backup that used previous keys or when importing configurations that were created with a previous key. Retain previous keys for these kinds of scenarios.

To delete an encryption key:

1. Confirm that the server is configured to use a different key as its primary.
   The tool cannot delete the primary key. View the `installed.properties` file and check that the value of the **encryption.keystore.alias** property is not the alias of the key that you are targeting for deletion. In an HA cluster, each member has its own `installed.properties` file, so each must be checked individually.
2. Update servers that use the targeted key.

   Edit the `installed.properties` and set the **`encryption.keystore.alias`** property to the alias of another key. In an HA cluster, each member has its own `installed.properties` file, so you must edit each member individually. For best results, set up the cluster so that all members use the same primary key.
3. Restart any updated servers.

   The server loads keys and the primary key setting only at startup. In an HA cluster, each updated member must be restarted.
4. Run the keydelete command without the --change-keystore argument to verify the action.
   In an HA cluster, all members share the same keystore file, so you must run the `keydelete` command on only one cluster member. For example, run the following command:

   ```
   keydelete --target-key <key>
   ```

   Here, the --target-key <key> flag is required. The <*key*> variable is the name of the key you are targeting for deletion. The key is not deleted unless `--change-keystore` is also specified.
5. Run the keydelete command again with the --change-keystore argument to complete the action.
   In an HA cluster, all members share the same keystore file, so you must run the **`keydelete`** command on only one cluster member:

   ```
   keydelete --target-key <key> --change-keystore
   ```

   Note that the `--change-keystore` argument is optional. It enables the modification of the keystore. Unless this flag is specified, the tool only demonstrates the change without completing the action.
6. Restart the server.

   The server never unloads a key. In an HA cluster, you must restart each member.

Authorization realms

Use the **Authorization Realms** pane to create authorization realms and user groups for the server. Groups can be imported from external systems, such as LDAP.

The available authorization realms for the server are as follows:

Internal Storage

Uses internal role management. The default authorization realm (Internal Security) is of this type.

LDAP or Active Directory

Uses external LDAP role management.

SSO

Provides single sign-on authorization.

OpenID Connect

Provides integration with Microsoft Entra ID for role-based authorization and Okta for group-based authorization.

Creating an internal storage authorization realm

Internal storage realms do not retrieve users from any external source. Instead, you add users to internal storage authorization realms manually.

1. On the server, click **Settings > Authorization (Groups) > Create Authorization Realm**. The **Create Authorization Realm** dialog box opens.
2. Enter a name in the **Name** field.
3. Ensure that **Internal Storage** is selected in the **Type** list.
4. Click **Save**.

Creating an LDAP authorization realm

An LDAP authorization realm defines how to use an external LDAP server for group authorization.

Before you begin

You must have the URL of the external LDAP server that you want to use for authorization.

About this task

An LDAP authorization realm defines how to search user groups. Typically, authorization realms work in tandem with LDAP authentication realms. When unknown users attempts to log on, an external LDAP server is used to authenticate them by using the authorization and authentication realms that you define. Administrators can also use the authorization and authentication realms to import users. Before importing users, make sure that authorization realm matches with the authentication realm unless there are two different LDAP systems being used. You might want to import users and assign them to teams before their first logon attempt. When you import users, you can supply additional parameters to further refine the search. For more information, see Importing LDAP users on page 2184.

To create an LDAP authorization realm, you identify the URL of the LDAP server, and define a role-based search. If you use an LDAP authentication realm, you do not need to define a user-based search. If you use an SSO authentication realm, you must also define a user-based search.

1. On the server, click **Settings > Authorization (Groups) > Create Authorization Realm**. The **Create Authorization Realm** dialog box opens.
2. In the **Name** field, enter a realm name.
3. In the **Type** list, select **LDAP or Active Directory**.
4. In the **LDAP URL** field, enter the URL of the LDAP or LDAPS server.
   Separate multiple servers by commas.
   Example
   For example, `ldap://ldap_server.my_domain.com:389,ldap://ldap_server.my_domain2.com:389,ldaps://ldaps_server.my_domain3.com:389`.
5. Define a role-based search by completing one of these steps:

   a. If you want to reference roles by members, select **Roles in LDAP reference their members; look up group membership by searching for roles**, and then specify the following parameters:

   Table 699. LDAP roles referenced by member properties

Field	Description
Group Search Base	The directory that is used for group searches, such as `ou=employees,dc=mydomain,dc=com`.
Group Search Filter	The LDAP filter expression that is used when you search for user entries. The user name replaces the *{1}* variable in the search pattern and the full user distinguished name (DN) replaces the *{0}* variable, for example, `(&(uniqueMember={0})(cn=BSO*))`.
Group Name	The name of the entry that contains the users' group names in the directory entries that are returned by the group search. If this entry is not specified, no group search runs. For example, enter `cn`.
Search Group Subtree	Subtrees (if any) are searched. If the item is not selected, the search is limited to the group search base.
Search For Containing Groups	Perform additional lookups to find indirectly assigned groups. This option may significantly increase the amount of time required to authenticate users. Consult your LDAP administrators before enabling this option.

   The values in the **Group Search Base** and **Search Group Subtree** fields define the scope of the search. Within the scope, groups that match the group search filter are searched. The value in the **Group Name** field specifies the LDAP attribute that contains the group name.

   b. If you want to reference roles by user attributes, select **User roles are defined as an attribute on that user; look up group membership using this attribute**, and then, in the **User Group Attribute** field, specify the attribute that contains group names in the user directory entry.
   If user groups are defined in LDAP as an attribute of the user, you must use the Group Attribute configuration.

6. **Optional:** Define a user-based search by completing one of these steps:

   Note: If you plan to use an SSO authentication realm with this authorization realm, you must define a user-based search. If you plan to use an LDAP authentication realm, you can skip this step.

   a. If user records exist in multiple directories, select **LDAP users may exist in many directories; search across LDAP using a criteria.**, and then specify these parameters:

   Table 700. LDAP user properties

Field	Description
User Search Base	The starting directory for the search, such as `ou=employees,dc=mydomain,dc=com`.
User Search Filter	The LDAP filter expression that is used when searching for user entries. The user name replaces the *{0}* variable in the search pattern, for example, `uid={0}`. If the value is not part of the DN pattern, wrap the value in parenthesis, such as `(sAMAccountName={0})`.
Search Group Subtree	Subtrees (if any) are searched. If the item is not selected, the search is limited to the search base.

   b. If users exist in a single directory, select **LDAP users exist in a single directory; use a pattern to create the DN for users**, and then, in the **User DN Pattern** field, specify the name.
   The name is substituted in place of *{0}* in the search pattern, such as `cn=[0],ou=employees,dc=mydomain,dc=com`.

7. **Optional:** Secure the communication between the LDAP server and the HCL DevOps Deploy (Deploy) server as described in Configuring SSL on Apache Tomcat and LDAP servers on page 2159.
8. **Optional:** Set the length of time for the server to wait for a response from the LDAP server. On the server, in the `installed.properties` file, set the property **com.sun.jndi.ldap.read.timeout** to the timeout period in milliseconds. Then, restart the server.
9. **Optional:** Enable **Nested Groups**. This feature enables Deploy to recursively search for roles in the LDAP server. **Note:** If a user logs in who has many direct LDAP group mappings, this feature could generate many extra LDAP queries.

   if a user logs in that has a lot of direct ldap group mappings

Results

The first time an unknown user attempts to log on, LDAP authorization realms are searched in an attempt to identify the user. If the user is found, a corresponding user ID is created in Deploy. In addition, if the user is part of an LDAP group, that group is imported too.

Creating an SSO authorization realm

A single sign-on (SSO) authorization realm uses an external server for authorization.

1. On the server, click **Settings > Authorization (Groups) > Create Authorization Realm**. The **Create Authorization Realm** dialog box opens.
2. Enter a name in the **Name** field.
3. Ensure that **SSO** is selected in the **Type** list.
4. Specify these parameters:

Table 701. SSO properties

Field	Description
Groups Header	The name of the SSO server parameter that contains the list of groups to which the user belongs.
Groups Delimiter	The string that delimits values that are found in the group header. Regular expression special characters must be escaped with the backslash (\) character.
Groups Regex	The regular expression that is used to parse the values in the group header. If specified, only the first matching group is used. If left unspecified, all values in the group header are used.

The values in the group header that match the search criteria are used to create HCL DevOps Deploy (Deploy) groups into which the user is placed.

Creating an OpenID Connect authorization realm

You can create an OpenID Connect (OIDC) authorization realm to use Microsoft Entra ID or Okta server for authorization.

1. On the server, click **Settings > Authorization (Groups) > Create Authorization Realm**. The **Create Authorization Realm** dialog box opens.
2. Enter a name in the **Name** field.
3. Ensure that **OIDC** is selected in the **Type** list.
4. Select any of the following vendors:

Table 702. OIDC Vendors

Vendor name	Description
Microsoft Entra ID	Allows you to retrieve group details from Microsoft Entra ID. When selected, specify any of the following group source: ◦ App Roles – To enable Application Role validation through the `roles` claim, the Microsoft Entra ID authorization realm requires specific configuration. You must ensure that the Entra ID manifest is set to `requestedAccessTokenVersion: 2` and that the application is exposed as an API. Finally, you must map the resulting API ID as a designated scope within the Deploy authentication realm settings.

Vendor name	Description
	◦ Security Groups – Microsoft Entra ID authorization realm calls the Microsoft Graph API to fetch Security Groups. The group's Object ID (GUID) is used to map to groups in Deploy. You must ensure that the Entra ID manifest is set to `requestedAccessTokenVersion: 2`.
Okta	To enable group-based authorization, the Okta realm validates the `groups` claim within the security token. This action requires a dedicated authorization server to be defined in Okta. Additionally, configure the application's authentication realm with the specific Issuer and associated endpoints provided by that authorization server.

5. Click **Save**.

Results

You have created an OIDC authorization realm with the selected vendor.

Creating a security group

You can create a group and grant permissions simultaneously to multiple users in a group. By grouping users, you can simplify the management of access control across various applications, environments, and resources.

About this task

You can create groups to authorize realms of the internal storage type. Other types, such as Lightweight Directory Access Protocol (LDAP) and import groups, do not allow manually created groups.

Note: Adding groups to LDAP and single sign-on (SSO) realms is deprecated.

1. Log in to Deploy.
2. Click **Settings > Authorization (Groups) > Groups > Create Group**.
   The **Create Group** dialog is displayed.
3. Type a name for the group in the **Name** field.
4. Optional: Type the description for the group in the **Description** field.
5. Select an authorization realm from the drop-down.

   Note: Groups are applicable only within the selected realm.

6. Click **Save**.

Results

You have created a group.

What to do next

You can add users to groups mannually. See, Adding users to groups manually on page 2178.

Related information

Creating an internal storage authorization realm on page 2173

Creating an LDAP authorization realm on page 2173

Creating an SSO authorization realm on page 2176

Adding users to groups manually

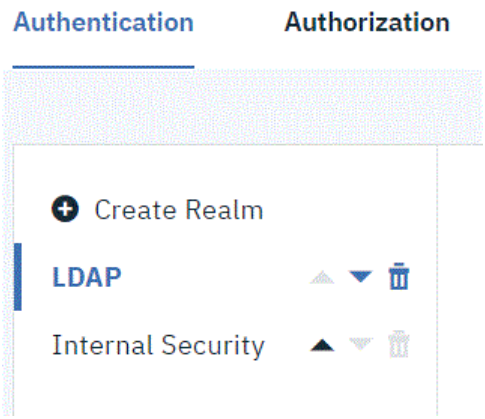
Authorization realms of the internal storage type can have manually created users. Others types, such as LDAP, import user groups cannot have manually created users.

1. On the server, click **Settings > Authorization (Groups) > Groups > Members > Add Member**. The **Add Group Member** dialog box opens.
2. Select the user from the **Group Member** list.
3. Click **Save**.

Authentication realms

Authentication realms manage users and determine user identity within authorization realms for the server.

Users can be created manually or imported from external systems, such as LDAP. Authentication is determined following the hierarchy of realms that is displayed on the **Authentication** tab. In the following figure, authentication is first determined in the Internal Security realm followed by the LDAP realm. A user who is listed in the LDAP realm can have different authorizations from those in the other realms.



If you have a number of authentication realms, you can reorder them by using the operation tools. Each realm can be moved up to a higher priority, moved down, or deleted by using the operation tools.

For information about authorization realms, see Authorization realms on page 2172.

Creating authentication realms

Use authentication realms to manage users and determine their identity within authorization realms.

About this task

To create an authentication realm on the server, specify where to store user information, such as internal storage on the server, a single sign-on service, an LDAP or Active Directory, or in OpenID Connect.

1. On the Web UI, click **Settings > Authentication (Users) > Create Realm**.
   The **Create Authentication Realm** window opens.
2. Enter a name and description for the realm.
3. Select the authentication realm type from the **Type** list.
   The available authentication realm types are: **LDAP or Active Directory** on page 2179, **Single Sign-on (SSO)** on page 2181, **Internal Storage**, and **OpenID Connect** on page 2181.
   If you select **Internal Storage**, no additional parameters are required.
   If you select **LDAP or Active Directory**, specify the following parameters:

Table 703. LDAP or Active Directory authentication realm properties

Field	Description
LDAP URL	The URL of the LDAP server that begins with `ldap://` or `ldaps://`. You can provide more than one URL, separated by provide more than one URL, the server tries the URLs one at a time until the server succeeds.
Authentication Mechanism	Indicates the authentication mechanism for use. The default mechanism is Simple. Select **DIGEST-MD5** to configu server to use DIGEST-MD5, and then specify the SASL realm.
Search Anonymously	When this checkbox is selected, LDAP accepts anonymous queries. If this check box is cleared, specify the LDAP s connection DN and associated password. This check box is selected by default.
Search Connection DN	Complete the LDAP distinguished name to search. This parameter is required if the **Search Anonymously** check bo
Search Connection Password	The password that is used for LDAP searches. This parameter is used with the value in the **Search Connection DN**
Specify how to search LDAP	Specifies how LDAP is searched. If users exist in multiple directories, select **LDAP users may exist in many directo across LDAP using a criterion**. Otherwise, select **LDAP users exist in a single directory; use a pattern to create the** Depending on the selection, more fields are displayed.
User Search Base	When you search multiple directories, specify the starting directory that is used for searches, such as `ou=employees,dc=mydomain,dc=com`.

Field		Description
	User Search Filter	The LDAP filter expression to use when you search for group entries, such as `(&(mail={0})(objectclass=ePerson)(sAMAccountName={0}))`. The user name replaces the *{1}* variable in the search p the full user DN replaces the *{0}* variable. If the value is not part of the DN pattern, enclose the value in parentheses `(accountName={0})`. For more information, see the help information for your LDAP server and look for information a user search filters.
	Search User Subtree	When you search multiple directories, select this check box to search directories that are subordinate to the base d
	User DN Pattern	When you search a single directory, the name is substituted in place of 0 in the pattern, for example, `cn={0},ou=employees,dc=yourcompany,dc=com`.
Name Attribute		Contains the user name in LDAP.
Email Attribute		Contains the user email address in LDAP.
Login Mechanism		Enables you to use OpenID Connect as an authentication mechanism with two modes: UID and alternate ID. For the properties that are common to both types, refer to the OpenID Connect authentication realm properties in th follows.
	Force Authentication through OpenID Connect	Prevents direct username and password authentication with LDAP for this realm.
	UID Claim	The OpenID Claim that contains the unique ID to use for LDAP lookup.
	Alternate ID Claim	The OpenID Claim that contains the alternate or custom ID to use for LDAP lookup.

Field		Description
	Alternate ID Search	The LDAP search filter that uses the alternate ID as `{0}`.

If you select **Single Sign-on**, specify the following parameters:

Table 704. SSO authentication realm properties

Field	Description
User Header Name	The header that contains the list of users.
Email Header Name	The header that contains the list of user email addresses.
Logout URL	The URL to which users are redirected after they log out of HCL DevOps Deploy (Deploy).

If you select **OpenID Connect**, specify the following parameters:

Table 705. OpenID Connect authentication realm properties

Field	Description
OpenID Provider Vendor	The vendor of the OpenID Provider. On selecting a vendor, the group membership is processed with Oper
OpenID Provider and Client Identity	
Client ID	The client ID that is registered with the OpenID provider.
Client Secret	The client secret that the OpenID provider issues upon registration.
Issuer	The issuer identifier for the issuer of the token response. Enter the issuer identifier, and click **Discover** to endpoint configuration automatically. The issuer identifier must not contain a query or fragment.
OpenID Endpoints	
JWKS URI	The JSON Web Key Sets (JWKS) URI endpoint.
Authorization Endpoint	The endpoint for authorization with the OpenID provider.
Token Endpoint	The endpoint for token requests with the OpenID provider.
Userinfo Endpoint	The endpoint for user information requests with the OpenID provider.
End Session Endpoint	The endpoint for RP-Initiated logout request (Deploy logout request). On logout, the Deploy user is redirected to the OpenID Provider logout page from which the user can log o Provider.
Custom OpenID Claims	
Username Claim	The claim that contains the username or account name of the user. Defaults to `sub`.

Field	Description
Name Claim	The custom claim that you want to use to display as a name in the Web UI. Defaults to `name`.
Email Claim	The custom claim that contains the user's e-mail ID and is displayed as email in the Web UI. Defaults to `e`
Additional Scopes	Any scope values that should be sent to the OpenID Provider in addition to the defaults openid, profile, ar
OpenID Options	
End session with ID token hint	Append the `id_token_hint` parameter to the end session endpoint URL. The effect of the hint is specific t provider. Some providers require the hint to successfully log out. Other providers alter their behavior with logout. Some providers may ignore it or some do not require it at all. Refer to the provider documentation

Notes:

- The Custom OpenID Claims and OpenID Options are not available in LDAP with OpenID Connect type authentication realm.
- Make sure that the external agent URL matches with the OpenID provider configuration for the redirect URI.
- The OpenID provider must be set to use RS256 as its token signature algorithm, so that Deploy communicates with it correctly.

4. Disable password-based login for the authentication realm by checking the **Disable Browser Login** option.

Notes:

- Administrators can still log in by using the login password via the admin URL, that is `https://hostname:port/admin`.
- Authentication tokens that are defined in this realm continue to work.
- Changing this setting does not disrupt active sessions.

5. Enter the number of user login attempts to accept in the **Allowed Login Attempts** field.
   A blank value means that an unlimited number of attempts is allowed. If a user exceeds the attempt limit, use the **Unlock User** action on the **Authentication** page to unlock the user.
6. Select the authorization realm from the **Authorization Realm** list.
   The internal security realm is available by default. See Authorization realms on page 2172.

Note: If you use an LDAP authentication realm with an LDAP authorization realm, the settings in this authentication realm override the settings in the LDAP authorization realm. If you use an SSO authentication realm with an LDAP authorization realm, you must specify connection settings on the LDAP authorization realm.

Results

The new user is created or imported, and listed on the **Authentication** page. When new users log on to the server and use their LDAP or SSO credentials, they are listed in the **Authentication** tab.

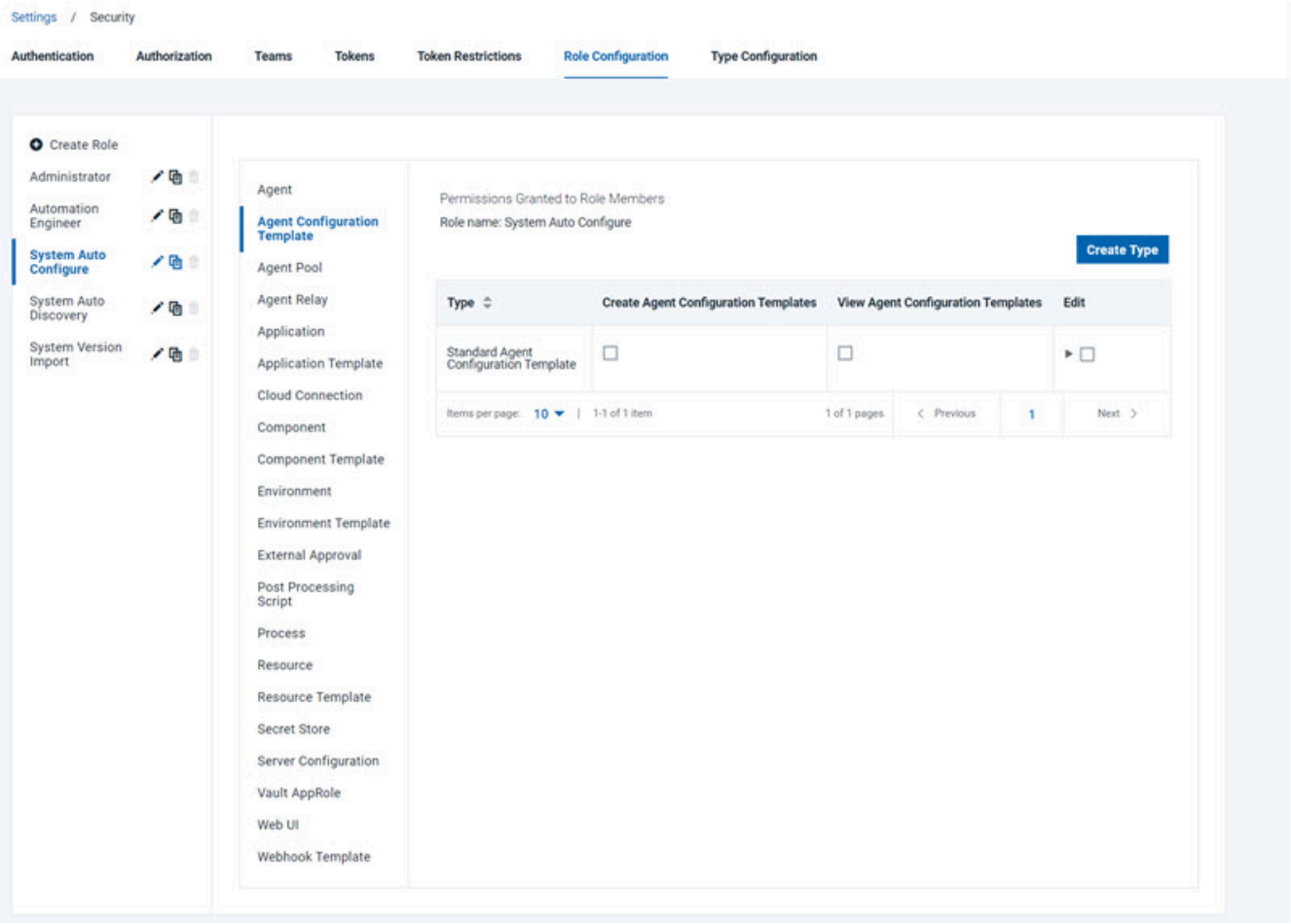
Note: In most cases, do not manage user passwords or remove users from the **Authentication** page. If an active user is removed from Deploy, the user can still log on to the server while their LDAP credentials are valid.

Creating users manually

You can manually create users on the server for internal security and SSO type authentication realms.

Before you begin

Editing user information requires permission to view the **Authentication (Users)**page. Make sure that the user's role has the **Manage Teams** and **WebUI | Settings Tab** permissions enabled.



1. On the server, click **Settings > Authentication (Users) > *target realm* > Create User**.
2. In the **Create User** dialog box, enter the user information:
    a. Enter the user name that is associated with the account in the **Username** field.
    b. Enter the password for the user in the **Password** field.
    Password complexity can be configured from the System Settings page.

c. Enter the user's name in the **Name** field.

d. Enter the email address that is associated with the account in the **Email** field.

3. Save your work when you are done.

What to do next

After you add a user to an authentication realm, you must also add the user to groups or teams.

Importing LDAP users

On the server, LDAP authentication realm users are imported from external LDAP systems.

About this task

LDAP authentication realms are searched whenever an unknown user attempts to log on to the server. If the user is found, a corresponding record is created in HCL DevOps Deploy (Deploy). Because LDAP users cannot be assigned to groups and teams until they have corresponding Deploy accounts, you can import users directly from LDAP systems.

1. On the server, click **Settings > Authentication (Users) > *selected LDAP realm* > Import User**.
2. In the **Import Users** dialog box, enter a name or pattern to search in the **Username or Pattern** field.
   The asterisk character (*) can be used as a wildcard in the pattern. If you use an asterisk by itself, you import all users for the selected LDAP realm.
3. Click **Save** to start the search.

Updating LDAP users and groups

LDAP users' name, addresses, and groups can be updated.

About this task

LDAP users and groups can be updated in following ways:

- **LDAP or Active Directory Authentication Realms** using LDAP authorization realms to update names and email addresses.
- **SSO authentication realms** using LDAP authorization realms to update groups. Names and email addresses are updated when user logs-in through SSO.

Note: You can not update users or groups if there is no LDAP Authorization realms being used.

1. On the server, click **Settings > Authentication (Users) > *select LDAP/SSO realm* > Update Users** for the LDAP authentication realm to use.
2. In the **Update All Users** dialog box, click **Submit** to update the LDAP user information for the selected LDAP or SSO realm.
   Result

The names, email addresses, and/or groups for all users in the realm are updated.

Note: User information is also synced when a user logs-in manually from the LDAP or Active Directory.

Editing user information manually

You can manually edit the stored name, email address, password, and additional settings for user accounts on the server for internal security realms. For an SSO authentication realm, name and email address are passed in via HTTP headers (for example, from an HTTP proxy that is translating Kerberos tokens into HTTP headers).

Before you begin

Editing user information requires permission to view the **Authentication (Users)** page. Make sure that you are assigned to an appropriate team and that the **Web UI | Settings Tab** permissions are granted.

Note: All of the following user configuration is optional and does not require to be performed in a particular order. You can make the changes as appropriate.

1. Go to **Settings > Authentication (Users)** and select an authentication realm.
2. In the **Actions** column, click **Edit** for the user account that requires updates.
    a. Edit the **Name** and **Email** fields as needed.
    b. Specify whether you want to disable automatic deletion of the user by using the **Exempt from Auto-Delete** option.
    c. Click **Save**.
3. Click **Delete** for the user that you want to delete.

    The user is removed from any groups and roles. This user may be re-added if this authentication realm is configured to create users automatically.
4. Click **Reset Password** to change the password for a user account.
    The **Reset User Password** window is displayed.
    a. Enter the new password in the **Password** field.
    b. Click **Save**.
5. Click **Lock User** for the user that you want to lock.
    The user is locked till the time you unlock it manually.

Results

The user information is updated.

Locking and deleting user accounts automatically

As part of your secured user account management process, you can configure the system to lock and delete idle or inactive users automatically.

Before you begin

You must have the following role permissions:

- **Web UI > Settings Tab**
- **Server Configuration > Edit Basic System Settings**
- **Server Configuration > View Users**

About this task

A user account is considered idle or inactive if the user has not logged in for a specific duration. You can configure the system to lock these accounts after a specific duration and further delete them, if neccessary.

1. On the Web UI, go to **Settings > System Settings > Security** settings.
2. Specify the number of days users must be inactive before their accounts are locked in the **User Idle Days to Lock** field.

   Note: You can leave the field empty or set the value to 0 to disable locking of the user accounts.

3. Specify the number of days users must be inactive before their accounts are deleted in the **User Idle Days to Delete** field, .
   The value must be greater than the value specified in the **User Idle Days to Lock field** or you can set the value to 0 to disable deletion of the user accounts.
4. Save the settings.

Results

You have configured automatic locking and deletion of user accounts in the system. The users are locked and deleted automatically based on your defined settings.

What to do next

At any point of time, you can click **Preview Idle User Lock** in the system settings to check the number of user accounts that are going to be locked and deleted. Additionally, you can view the Last Logon Time, Locked Time, Will Lock, and Will delete times of each user in the **Authentication (Users)** page.

Enforcing a maximum login duration

You can enforce a maximum login duration to require users to re-authenticate after a specific period, regardless of the user activity. This setting is useful for organizations whose security policies prohibit persistent login sessions.

Before you begin

You must have the following role permissions:

- **Web UI > Settings Tab**
- **Server Configuration > Edit Basic System Settings**
- **Server Configuration > View Users**

About this task

The **Maximum Login Time** setting overrides the **Remember me** configuration. After this setting is active, a countdown timer is displayed in the log-off menu. The system provides the following visual warnings:

- A black banner appears when the session time remaining reaches 30 minutes.
- The message changes to red when the time remaining drops below 5 minutes.

1. In the Web UI, navigate to **Settings > System Settings > Security** settings.
2. In the **Maximum Login time (hours)** field, specify the number of hours users may remain logged in.

   Note: If the value is set to 0 or a negative number, the maximum login time enforcement is disabled.

3. Save the settings.
4. Log out and log back into the Web UI.

   Note: Changes to this setting only apply to new sessions. Existing active sessions remain unchanged until the user logs out and back in.

Results

The maximum login time is enforced. When the timer reaches zero, the following actions occur:

- The server terminates the session immediately.
- The client triggers the logout endpoint and refreshes to the login screen.
- OIDC and single sign-on (SSO) sessions are redirected for full de-authentication.

Roles and permissions

A *role* is a list of permissions that users have on the HCL DevOps Deploy (Deploy) server. These permissions include the ability to create, edit, and delete objects, such as applications, environments, and components. The permission also includes the parts of the server interface that users can access.

When you assign users to a role, they get all permissions that are granted to the role. However, these permissions apply only to the team on which the user has the role. For example, if a user has the Developer role on team A, that user has the role permissions on the environments, applications, and components that are assigned to team A. The user does not have any permissions on the environments, applications, and components that are assigned to team B unless you assign that user a role on team B.

You can assign permissions to a user only by giving that user a role on a team. You cannot assign a user to a role without also putting that user on a team. Similarly, you cannot assign a user to a team without giving that user a role on the team.

Typically, the permissions that are granted to a role define a particular activity that a user might do, such as running deployments. Users can be assigned to more than one role, and users can have different roles on different teams. You decide the number of roles and their permissions that you assign to users. Initially, Deploy provides one role, the Administrator role, which has all possible permissions granted to it.

Note: If a role grants the Create permission for some object type, such as components, it is important that the role also grant the Edit Basic Settings and Manage Teams permissions for that type. Otherwise, the options for creating the object are limited.

The Create permission is different from other permission types. The create type is not scoped to a specific resource but is more broadly scoped. Team members with a create-type permission can create resources of the related type without team members first being attached to a specific resource. To continue the environment example, users with the create environment permission can create environments even if their team is not attached to a specific environment.

Most objects have the create, view, and edit permissions that most object types. Environments and processes have an execute permission. This permission controls the ability to run processes.

Creating roles and assigning permissions

You can create roles on the server and assign permissions to them.

About this task

By default, new roles have no permissions. Therefore, when you create a role, you must add all the permissions for that role. For example, if you do not add permission to view the pages of the server user interface in the **Web UI** section, users can log in to the server but cannot see any information.

Several default roles are available. These roles have matching default users.

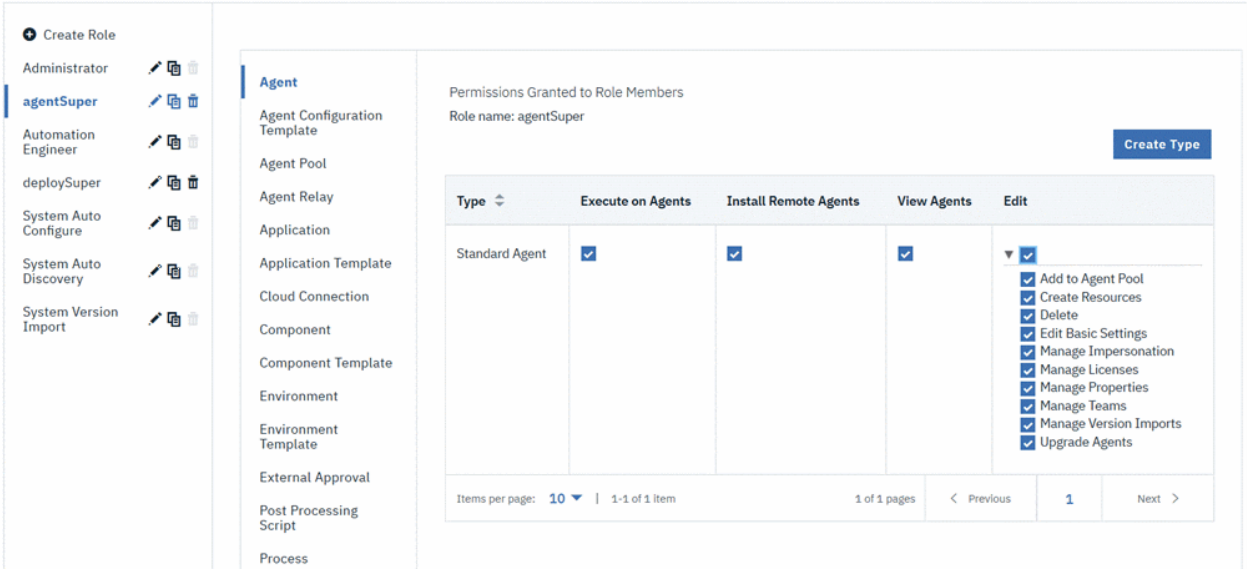
- **Administrator** role has all permissions. It is appropriate for administrators.
- **Automation Engineer** is an example of a role for a typical non-adminstrator user. It includes permissions to create and edit most things on the server, including applications, component, and processes, but not the permission to delete things, change permissions, or change system settings.
- **System Auto Configure** role has the permissions to create resources.
- **System Auto Discovery** role is used by plug-ins that discover information about systems, such as the WebSphere® Application Server - Deploy plug-in.
- **System Version Import** role has only the permission to manage versions.

Create roles and assign them permissions as follows:

1. On the server, click **Settings > Role Configuration**.
2. In the **Role Configuration** pane, click **Create Role**.
3. Enter a name for the role, and then click **Save**.
4. Click the role to select it.
5. With the role selected, select the object type to which you want to assign permissions, such as environments, applications, or components.
   All security types for the selected object type are displayed. For information about creating security types, see Creating security types on page 2221.

In the following figure, the security types for the **Agent** product area are displayed. Because the `agentSuper` role is selected, only that role is affected by permission changes to the `Standard Agent` and `agentManager` security types. That is, if you select the **Agent** object type for the `deploySuper` role, the granted permissions reflect the settings that were defined the last time that role was modified.

Tip: Some users might no be unable to see agent relays in the user interface, even though they were visible before the upgrade. You can set the permission for these users to view agent relays: Click **Agent Relay** in the permissions area list, and then select **View Agent Relays**.



6. In the **Permissions Granted to Role Members** pane of an object type, grant permissions by selecting the associated checkbox for a security type.
   You can select all edit subpermissions by selecting the **Edit** check box. To deselect all subpermissions, clear the corresponding check boxes. Select individual subpermissions by expanding the **Edit** list and clicking the corresponding check boxes. For information about editing subpermissions, see Security types on page 2219

   Note: If a role grants the Create permission for some object type, such as components, it is important that the role also grant the Edit Basic Settings and Manage Teams permissions for that type. Otherwise, the options for creating the object are limited.

7. **Required:** Click the **Web UI** category, and specify the pages in the server user interface that the role can access. Each option in this list refers to a top-level page in the user interface.
   Example
   For example, if you do not select the **Applications Tab** option in this list, users with this role do not see the top-level **Applications** tab when they log in to the server. You can select each tab separately.
8. **Optional:** If the role users need system permissions, such as the ability to change permissions, click **Server Configuration**, and specify those permissions.

For more information, see Setting server configuration security on page 2190.

9. **Optional:** Create a role from an existing role by clicking icon next to the role you want to copy.
   Copying a role helps you save time in creating a similar role. The copied role duplicates the permission set of the existing role to allow to make minor modifications.

What to do next

Assign roles to users by adding users or groups to teams. You cannot assign a role to a user directly; users must be assigned roles when they are added to teams. See Assigning users to teams on page 2214.

Setting server configuration security

Set the server configuration permissions to define what users can do with the HCL DevOps Deploy (Deploy) server security, also referred to as system security. Typically, these permissions are granted to administrative roles.

The following server configuration permissions are available:

Table 706. Server configuration permissions

Permission	Description
Add Team Members	Add users to roles that have the same or fewer permissions than those of the granting user.
Bulk Manage Cleanup	Allows users to view, and edit the cleanup settings of multiple components and environments from the **Cleanup Configuration** page.
Create with Team Assignment	Grants users the ability to add any additional team and type combination to an object if they have the Create permission on atleast one team/type mapping.
Create Tags	Grants users the permission to create tags. Users can create tags in the **Tags** page with only this permission. To create tags on an object, users must have the **Edit Basic Settings** permission on the object type. See Creating tags on page 2520.
Edit Basic System Settings	Enables users to edit the options on the **System Settings** page. See, System settings on page 2086. This activity is written to the audit log.
Edit Network Settings	Grants users the permission to access the **Network** page and manage servers in a server cluster.
Edit Tags	Grants users the permission to edit tags. Users can edit tags in the **Tags** page. Users can apply tags to an object only if they have the **Edit Basic Settings** permission enabled on the object type. See Adding tags to objects on page 2521.
Manage Audit Log	Grants users the permission to manually cleanup the audit log, and configure automatic cleanup settings. If you have this permission but not the edit basic

Table 706. Server configuration permissions (continued)

Permission	Description
	settings permission, you can edit the audit cleanup settings from the **Audit Log** page. This activity is written to the audit log.
Manage Auth Token Restrictions	Create, modify and delete token restrictions. See, Restricting authentication tokens on page 2223
Manage Diagnostics	Enable users to access the diagnostics information including the Rest Call Log, Java Thread dumps, and Metadata indexing. This activity is written to the audit log.
Manage Java Packages	Allows users to add and delete Java packages.
Manage Logging Settings	Grants users the permission to view the **Logging** page and edit the log4j configuration. This activity is written to the audit log.
Manage Maintenance Mode	Allows users to enable maintenance mode.
Manage Notifications Schemes	Enable users to manage notification schemes used for notification emails. See, Creating Notifications in a Notification Scheme on page 2081. This activity is written to the audit log.
Manage Plugins	Grants users the permission to install new plug-ins; see Installing plug-ins from the plug-in site on page 2559. Install and delete activities are written to the audit log.
Manage Resource Roles	Grants users the permission to create and delete resource roles, which are created by some plug-ins. In most cases, you do not need to add or change resource roles because they are used internally.
Manage Security	Grants users the permission to manage security configuration, including roles, authentication realms, authorization realms, and tokens. Users without this permission cannot access or change the security functions. This activity is written to the audit log.
Manage Server Keystore	Grants users the permission to add and remove server keystore entries.
Manage Statuses	Grants users the permission to create and edit version, inventory, and snapshot statuses.
Manage System	Grants users permission to view and manage options on the **System Settings** page. See System settings on page 2086. This activity is written to the audit log.
Manage System Properties	Grants users the permission to create and edit system properties. This activity is written to the audit log.
Read Artifact Set List	Enables agent relays to use component version replication by status. Assign a user with this permission to an authentication token intended for an agent relay.
Read Z Inventory	Grants users the permission to perform z/OS inventory search filter.

Table 706. Server configuration permissions (continued)

Permission	Description
Release Locks	Enables users to manually release any locks currently held. See, Managing locks on page 2084. This action is written to the audit log.
Run Deployment Triggers	Grants users permission to run the deployment from deployment triggers. The user will need to have the required permissions and roles as if they were executing the deployment manually.
Run Processes in Maintenance Mode	Allows users to execute a new deployment while the maintenance mode is enabled.
Use AI	Allows users to perform AI analysis on failed deployments.
View Audit Log	Grants users the permission to view the **Audit Log** page and download log files. This activity is written to the audit log.
View Basic System Settings	Enables users to view the options on the **System Settings** page. See, System settings on page 2086.
View Bulk Reports	Enables users to view bulk reports.
View Locks	Enables users to view any locks currently held. See, Managing locks on page 2084. This action is written to the audit log.
View Network Settings	Grants users the permission to access the **Network** page.
View Output Log	Grants users the permission to view the **Output Log** page and download log files. This activity is written to the audit log.
View Security	Grants users the permission to view the security configuration including the **Role Configuration**, **Authentication Settings**, **Authorization**, **Teams**, **Token Restrictions**, and **Type Configuration**. Users with this permission have read-only access and cannot modify the security functions. This activity is written to the audit log.
View Users	Enables users to view users activity.

To add system security permissions to roles:

1. Click **Settings > Role Configuration > *selected role* > Server Configuration** to open the **Permissions Granted to Role Members** page for the target role.
2. Use the toggle button to add appropriate permissions to the role.

Permission reference

This table lists the permissions on the server that you can assign to roles.

The following table lists the product areas with their available permissions. Because of the large number of edit sub-permissions, they are provided in a separate column.

Table 707. Permissions

Product area	Available permissions	Subpermissions
Agent	• **Execute on agents**: run processes on the agent • **Install Remote Agents** enable you to install remote agents. • **View Agents**: access agents in the UI but cannot modify them • **Edit**: modify and delete agents	Edit • **Add to Agent Pool**: assign agents to agent pools • **Create Resources**: create agent-type resources • **Delete**: delete agents • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Impersonation**: set up user impersonation for agents • **Manage Licenses**: set agent license type • **Manage Properties**: create and modify agent properties • **Manage Teams**: assign agents to teams • **Manage Version Imports**: import component versions with the agent • **Upgrade Agents**: upgrade agents
Agent Configuration Template	• **Create Agent Configuration Templates**: create agent configuration templates • **View Agent Configuration Templates**: access agent configuration template in the UI but cannot modify them • **Edit**: modify and delete agent configuration templates	Edit • **Delete**: delete agent configuration templates • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign agent configuration templates to teams
Agent pool	• **Create Agent Pools**: create agent pools • **View Agent Pools**: access agent pools in the UI but cannot modify them • **Edit**: modify and delete agent pools	Edit • **Create Resources**: create agent pool resources • **Delete**: delete agent pools • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Pool Members**: change the agents that are in a pool • **Manage Teams**: assign agent pools to teams

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
Agent relay	• **View Agent Relays**: see the agent relays that are listed on the **Resources** tab	Edit • **Delete**: delete agent relays • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign agent relays to teams
Application	• **View Applications**: access applications in the UI but cannot modify them • **Create**: create applications • **Edit**: modify and delete applications	Create • **Create Applications**: create applications without using a template • **Create Applications From Template**: create applications by using templates to define their attributes Edit • **Delete**: delete applications • **Delete Snapshot**: delete snapshot • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Blueprints**: create and modify blueprints • **Manage Components**: add and remove components • **Manage Environments**: create and modify environments • **Manage Pipelines**: create and modify pipelines • **Manage Processes**: create application processes • **Manage Properties**: create and modify application properties • **Manage Snapshots**: create and modify snapshots • **Manage Teams**: assign teams to applications

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
		• **Manage Webhooks**: create and modify webhooks • **Run Component Processes**: run processes that are defined for components that are attached to the application
Application template	• **Create Application Templates** • **View Application Templates** • **Edit**	Edit • **Delete**: delete applications • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Blueprints**: create and modify blueprints • **Manage Component Tags**: add and remove component tags • **Manage Environment Templages**: create and modify environment templates • **Manage Processes**: create application processes • **Manage Properties**: create and modify application properties • **Manage Teams**: assign teams to applications
Cloud connection	• **Create Cloud Connections**: create connections • **View Cloud Connections**: access connections in the UI but cannot modify them • **Edit**: modify and delete connections	Edit • **Delete**: delete cloud connections • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to cloud connections
Component	• **View Components**: access components in the UI but cannot modify them • **Create**: create components • **Edit**: modify and delete components	Create • **Create Components**: create components without using a template • **Create Components From Template**: create components by using templates to define their attributes

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
		Edit • **Approve Promotion**: approve component version promotion • **Delete**: delete components • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Component Cleanup**: modify component cleaup settings • **Manage Configuration Templates**: create configuration templates • **Manage Process Lock**: create process lock • **Manage Processes**: create component processes • **Manage Properties**: create component properties • **Manage Teams**: assign teams to components • **Manage Version Status**: create component version status • **Manage Versions**: import component versions
Component template	• **Create Component Templates**: create component templates • **View Component Templates**: access component templates in the UI but cannot modify them • **Edit**: modify and delete component templates	Edit • **Approve Promotion**: approve component template promotion • **Delete**: delete component templates • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Process Lock**: create component template process lock • **Manage Processes**: create and modify component template processes • **Manage Properties**: create and modify component template properties • **Manage Teams**: assign teams to component templates

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
Environment	• **Execute on Environments**: run parent applications • **View Environments**: access environments in the UI but not modify them • **Create**: create environments • **Edit**: modify and delete environments	Create • **Create Environments**: create environments without using a template • **Create Enviornments From Template**: create environments by using templates to define their attributes Edit • **Delete**: delete environments • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Approval Processes**: create and modify approvals and notifications • **Manage Base Resources**: add base resources to environments • **Manage Deployment Triggers**: create and modify deployment triggers. • **Manage Environment Cleanup**: modify environment cleaup settings • **Manage Properties**: create and modify environment properties • **Manage Teams**: assign teams to environments
Environment template	• **Create Environment Templates** • **View Environment Templates** • **Edit**	Edit • **Delete**: delete applications • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Approval Processes**: create modify approvals and notifications • **Manage Environment Template Cleanup**: modify environment template cleaup settings

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
		• **Manage Properties**: create and modify environment properties • **Manage Teams**: assign teams to environments
External Approval	• **Create External Approvals:** create external approvals • **View External Approvals:** access external approval definitions in the UI but cannot modify them • **Edit:** modify and delete external approvals	Edit • **Delete**: delete external approvals • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Properties**: create and modify external approval properties • **Manage Teams**: assign teams to external approvals
Post Processing Script	• **Create Post Processing Script:** create post processing scripts • **View Post Processing Script:** access post processing script definitions in the UI but cannot modify them • **Edit:** modify and delete post processing script	Edit • **Delete**: delete post processing script • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to post processing script
Process	• **Create Processes**: create processes • **Execute Processes**: run processes • **View Processes**: access process definitions in the UI but cannot modify them • **Edit**: modify and delete processes	Edit • **Delete**: delete generic processes • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Properties**: create and modify generic process properties • **Manage Teams**: assign teams to generic processes • **Manage Webhooks**: create and modify webhooks

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
Resource	• **Create Resources**: create resources • **Execute on Resources**: Run processes on agent resources • **View Resources**: access resources in the UI but cannot modify them • **Edit**: modify and delete resources	Edit • **Delete**: delete resources • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Children**: create, modify, and delete resource hierarchies • **Manage Impersonation**: set up user impersonation for agent and agent pool resources • **Manage Properties**: create and modify resource properties • **Manage Teams**: assign teams to resources • **Map to Environments**: assign resources to application environments
Resource template	• **Create Resource Templates**: create resource templates • **View Resource Templates**: access resource templates in the UI but cannot modify them • **Edit**: modify and delete resource templates	Edit • **Delete**: delete resource templates • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Resources**: create, modify, and delete resource hierarchies • **Manage Teams**: assign teams to resource templates
Secret Store	• **Create Secret Stores**: create secret stores • **View Secret Stores**: access Secret Stores in the UI but cannot modify them • **Edit**: modify and delete secret stores	Edit • **Delete**: delete secret stores • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to secret stores

Table 707. Permissions (continued)

Product area	Available permissions	Subpermissions
Vault AppRole	• **Create Vault AppRoles**: create Vault AppRoles • **View Vault AppRoles**: access Vault AppRoles in the UI but cannot modify them • **Edit**: modify and delete Vault AppRoles	Edit • **Delete**: delete Vault AppRoles • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to Vault AppRoles
Vault Certificate	• **Create Vault Certificates**: create Vault certificate • **View Vault Certificates**: access Vault certificates in the UI but cannot modify them • **Edit**: modify and delete Vault certificates	Edit • **Delete**: delete Vault certificates • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to Vault certificates
Vault Ldap User	• **Create Vault Ldap Users**: create Vault LDAP users • **View Vault Ldap Users**: access Vault LDAP users in the UI but cannot modify them • **Edit**: modify and delete Vault LDAP users	Edit • **Delete**: delete Vault LDAP users • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to Vault LDAP users
Webhook Template	• **Create Webhook Templates**: create webhook templates • **View Webhook Templates**: access webhook templates in the UI but cannot modify them • **Edit**: modify and delete webhook templates	Edit • **Delete**: delete webhook templates • **Edit Basic Settings**: edit configuration settings on the **Basic Settings** panel • **Manage Teams**: assign teams to webhook templates

The page of permissions also includes **Server Configuration** and **Web UI** sections. These permissions affect how users can modify security settings on the server and which pages in the server user interface they can see.

Table 708. Server configuration permissions

Permission	Description
Add Team Members	Add users to roles that have the same or fewer permissions than those of the granting user.
Bulk Manage Cleanup	Allows users to view, and edit the cleanup settings of multiple components and environments from the **Cleanup Configuration** page.
Create with Team Assignment	Grants users the ability to add any additional team and type combination to an object if they have the Create permission on atleast one team/type mapping.
Create Tags	Grants users the permission to create tags. Users can create tags in the **Tags** page with only this permission. To create tags on an object, users must have the **Edit Basic Settings** permission on the object type. See Creating tags on page 2520.
Edit Basic System Settings	Enables users to edit the options on the **System Settings** page. See, System settings on page 2086. This activity is written to the audit log.
Edit Network Settings	Grants users the permission to access the **Network** page and manage servers in a server cluster.
Edit Tags	Grants users the permission to edit tags. Users can edit tags in the **Tags** page. Users can apply tags to an object only if they have the **Edit Basic Settings** permission enabled on the object type. See Adding tags to objects on page 2521.
Manage Audit Log	Grants users the permission to manually cleanup the audit log, and configure automatic cleanup settings. If you have this permission but not the edit basic settings permission, you can edit the audit cleanup settings from the **Audit Log** page. This activity is written to the audit log.
Manage Auth Token Restrictions	Create, modify and delete token restrictions. See, Restricting authentication tokens on page 2223
Manage Diagnostics	Enable users to access the diagnostics information including the Rest Call Log, Java Thread dumps, and Metadata indexing. This activity is written to the audit log.
Manage Java Packages	Allows users to add and delete Java packages.
Manage Logging Settings	Grants users the permission to view the **Logging** page and edit the log4j configuration. This activity is written to the audit log.
Manage Maintenance Mode	Allows users to enable maintenance mode.

Table 708. Server configuration permissions (continued)

Permission	Description
Manage Notifications Schemes	Enable users to manage notification schemes used for notification emails. See, Creating Notifications in a Notification Scheme on page 2081. This activity is written to the audit log.
Manage Plugins	Grants users the permission to install new plug-ins; see Installing plug-ins from the plug-in site on page 2559. Install and delete activities are written to the audit log.
Manage Resource Roles	Grants users the permission to create and delete resource roles, which are created by some plug-ins. In most cases, you do not need to add or change resource roles because they are used internally.
Manage Security	Grants users the permission to manage security configuration, including roles, authentication realms, authorization realms, and tokens. Users without this permission cannot access or change the security functions. This activity is written to the audit log.
Manage Server Keystore	Grants users the permission to add and remove server keystore entries.
Manage Statuses	Grants users the permission to create and edit version, inventory, and snapshot statuses.
Manage System	Grants users permission to view and manage options on the **System Settings** page. See System settings on page 2086. This activity is written to the audit log.
Manage System Properties	Grants users the permission to create and edit system properties. This activity is written to the audit log.
Read Artifact Set List	Enables agent relays to use component version replication by status. Assign a user with this permission to an authentication token intended for an agent relay.
Read Z Inventory	Grants users the permission to perform z/OS inventory search filter.
Release Locks	Enables users to manually release any locks currently held. See, Managing locks on page 2084. This action is written to the audit log.
Run Deployment Triggers	Grants users permission to run the deployment from deployment triggers. The user will need to have the required permissions and roles as if they were executing the deployment manually.
Run Processes in Maintenance Mode	Allows users to execute a new deployment while the maintenance mode is enabled.
Use AI	Allows users to perform AI analysis on failed deployments.
View Audit Log	Grants users the permission to view the **Audit Log** page and download log files. This activity is written to the audit log.
View Basic System Settings	Enables users to view the options on the **System Settings** page. See, System settings on page 2086.

Table 708. Server configuration permissions (continued)

Permission	Description
View Bulk Reports	Enables users to view bulk reports.
View Locks	Enables users to view any locks currently held. See, Managing locks on page 2084. This action is written to the audit log.
View Network Settings	Grants users the permission to access the **Network** page.
View Output Log	Grants users the permission to view the **Output Log** page and download log files. This activity is written to the audit log.
View Security	Grants users the permission to view the security configuration including the **Role Configuration**, **Authentication Settings**, **Authorization**, **Teams**, **Token Restrictions**, and **Type Configuration**. Users with this permission have read-only access and cannot modify the security functions. This activity is written to the audit log.
View Users	Enables users to view users activity.

Table 709. Web UI permissions

Permission	Description
Applications Tab	Access to the **Applications** tab. See Applications on page 2344.
Components Tab	Access to the **Components** tab. See Components on page 2247.
Configuration Tab	Access to the **Configuration** tab. See Creating, setting, and editing properties on page 2504.
Dashboard Tab	Access to the **Dashboard** tab. See Dashboard on page 2245.
Deployment Calendar Tab	Access to the **Calendar** tab; see Scheduling deployments on page 2545.
Processes Tab	Access to the **Processes** tab; see Generic processes on page 2451.
Reports Tab	Access to the **Reports** tab; see Reporting on page 2547.
Resources Tab	Access to the **Resources** tab; see Resources on page 2312.
Secret Store Tab	Access to the **Secret Store** tab; see Secret stores on page 2206.
Settings Tab	Access to the **Settings** tab; see Server settings and configuration on page 2069.
Work Items Tab	Access to the **Work Items** tab; see Responding to approval requests on page 2394.

Permissions reference for using application templates

Permissions for roles that create applications from templates.

Application templates can contain information about the components and environments that are initially available to applications that are created from the templates. The composition of the application template determines which role permissions are needed to create an application from a template.

Note: The role permissions in the following tables are only the permissions that are required to create the application from a template. More permissions are necessary to add elements to an application, run processes, or complete a deployment.

To create an application from a template, the user's role must have the following minimum permissions.

Note: The user cannot select components or create environments for the application.

Table 710. Permissions for role that creates applications from an application template

Product area	Permissions	Subpermissions
Application	• **View Applications**	Edit • **Edit Basic Settings** • **Manage Properties** • **Manage Teams** Create • **Create Applications From Template**
Application Template	• **View Application Templates**	
Resource	• **Create Resources**	Edit • **Manage Teams**

To assign existing components to an application from the application template wizard, the user's role must also have the following permissions.

Note: To create an application from the application template wizard, the user's role must also have the permissions that are listed in Table 1.

Table 711. More permissions that are required to specify components

Product area	Permissions	Subpermissions
Application		Edit • **Manage Components**
Component	• **View Components**	

To create environments for an application from the application template wizard, the user's role must also have the following permissions. To map resources to the environments from the application wizard, the role must also have the permissions in Table 2.

Note: To create an application from the application template wizard, the user's role must also have the permissions that are listed in Table 1.

Table 712. Permissions to create environments and map resources to them

Product area	Permissions	Subpermissions
Agent	• **View Agents**	
Application		Edit • **Manage Approvals** • **Manage Environments**
Environment	• **View Environments**	Edit • **Edit Basic Settings** • **Manage Base Resources** • **Manage Properties** Create • **Create Environments From Template**

Table 712. Permissions to create environments and map resources to them (continued)

Product area	Permissions	Subpermissions
Environment Template	• **View Environment Templates**	
Resource	• **View Resources**	Edit • **Manage Children** • **Map to Environments** • **Manage Teams**
Resource Template	• **View Resource Templates**	

Secret stores

You can store user credentials in Hashicorp Vault and integrate with HCL DevOps Deploy (Deploy) to retrieve values from the Vault.

In Deploy, you store passwords as secured properties. For example, the database credentials. These passwords are stored in the database. Instead of storing the passwords in the database, you can store them in HashiCorp Vault. HashiCorp Vault is a third-party tool that stores and controls access to passwords and other secrets critical in modern computing.

The Deploy secret store enables you to retrieve user credentials during deployment without having stored the passwords in the database.

Deploy uses the following Vault authentication methods:

- **App Roles** - A role-based authentication mechanism where you have a defined set of access. AppRole uses RoleID and SecretID as the master authentication mechanism which allows Deploy to get the passwords that an AppRole can access. For information about the Vault AppRole, refer to the Vault AppRole auth documentation.
- **LDAP Users** - An LDAP-based authentication mechanism to call LDAP users from the Vault secret store. For information about the Vault LDAP, refer to the Vault LDAP auth documentation.
- **Certificates** - An SSL/TLS client certificate-based authentication mechanism to call certificate from the Vault secret store. For information about the Vault TLS certificate, refer to the Vault TLS certificate auth documentation.

The Deploy secret store is different from the Vault secret store. Deploy secret stores can have multiple secret stores. Each secret store in Deploy can connect to a different Vault server.

Adding Vault AppRoles to the Deploy secret store

You can add Vault AppRoles users to the HCL DevOps Deploy (Deploy) secret store to authenticate users with Vault-defined AppRole.

Before you begin

- Ensure that you have installed the Vault server and obtained the server address.
- You must have the following permissions:
  - Create, view, and edit secret store permissions to create a secret store.
  - Create, view, and edit Vault AppRole permissions to manage AppRoles of a secret store.

1. Click **Create Vault Secret Store**.
2. Create a secret store in the **Manage Vault Secret Store** window and provide the following details:

Parameter	Description
Name	Identifies the Vault secret store name. **Note:** Characters other than `a-z`, `A-Z`, `0-9`, and `-` are not allowed.
Description	Conveys more information about the Vault.
Teams	Allows access to the Vault secret store to specific teams.
URL	Indicates the Vault server address.

3. Save your changes.
   The secret store lists on the **Secret Stores** page.
4. Add AppRoles users to the secret store.
   a. Click the secret store to which you want to add the AppRole.
   b. Click **App Roles**.
   c. Click **Create Vault AppRole**.

      The **Manage Vault AppRole** window is displayed.

   d. In the **Manage Vault AppRole** window, enter the following AppRole details:

Parameter	Description
Name	Identifies the Vault AppRole name. **Note:** Characters other than `a-z`, `A-Z`, `0-9`, and `-` are not allowed.
AppRole Id	Specifies the RoleID of the Vault AppRole.
Secret Id	Specifies the SecretID of the Vault AppRole.
Teams	Allows access of the AppRole to specific teams.

e. Save your changes.

The AppRole is added to the Vault AppRoles page.

Results

You have added AppRole users to the Deploy secret store.

What to do next

You can test the integration. See Testing Vault AppRole integration on page 2211.

Adding Vault LDAP users to the Deploy secret store

You can add Vault LDAP users to the HCL DevOps Deploy (Deploy) secret store to authenticate users with Vault-defined LDAP.

Before you begin

- Ensure that you have installed the Vault server and obtained the server address.
- You must have the following permissions:
  - Create, view, and edit secret store permissions to create a secret store.
  - Create, view, and edit Vault Ldap User permissions to manage LDAP users of a secret store.

1. Click **Create Vault Secret Store**.
2. Create a secret store in the **Manage Vault Secret Store** window and provide the following details:

Parameter	Description
Name	Identifies the Vault secret store name. **Note:** Characters other than `a-z`, `A-Z`, `0-9`, and `-` are not allowed.
Description	Conveys more information about the Vault.
Teams	Allows access to the Vault secret store to specific teams.
URL	Indicates the Vault server address.

3. Save your changes.
   The secret store lists on the **Secret Stores** page.
4. Add LDAP users to the secret store.
   a. Click the secret store to which you want to add the LDAP users.
   b. Click **Ldap Users**.
   c. Click **Create Vault Ldap User**.

   The **Manage Vault Ldap User** window is displayed.

d. In the **Manage Vault Ldap User** window, enter the following LDAP user details:

Parameter	Description
Name	Identifies the Vault LDAP user name.
Password	Specifies the password of the LDAP user.
Teams	Allows access of the LDAP user to specific teams.

Note: The LDAP user name field does not allow characters other than `a-z`, `A-Z`, `0-9`, and `-`.

e. Save your changes.

The LDAP user is added to the Vault LDAP page.

Results

You have added LDAP users to the Deploy secret store.

What to do next

You can test the integration. See Testing Vault LDAP integration on page 2211.

Adding Vault certificate to the Deploy secret store

You can add the Vault certificates to the HCL DevOps Deploy (Deploy) secret store to authenticate with Vault-defined SSL/TLS client certificate.

Before you begin

- You must have completed the following tasks:
  - Installed the Vault server and obtained the server address.
  - Ensured that an SSL/TLS client certificate is available in the PEM format.
- You must have the following permissions:
  - Create, view, and edit secret store permissions to create a secret store.
  - Create, view, and edit Vault Certificate permissions to manage certificate of a secret store.

1. Click **Create Vault Secret Store**.
2. Create a secret store in the **Manage Vault Secret Store** window and provide the following details:

Parameter	Description
Name	Identifies the Vault secret store name.

Parameter	Description
	Note: Characters other than `a-z`, `A-Z`, `0-9`, and `-` are not allowed.
Description	Conveys more information about the Vault.
Teams	Allows access to the Vault secret store to specific teams.
URL	Indicates the Vault server address.

3. Save your changes.
   The secret store lists on the **Secret Stores** page.
4. Add certificate to the secret store.
   a. Click the secret store to which you want to add the certificate.
   b. Click **Certificates**.
   c. Click **Create Vault Certificate**.

      The **Manage Vault Certificate** window is displayed.

   d. In the **Manage Vault Certificate** window, provide the following details:

Parameter	Description
Name	Identifies the Vault certificate name. The certificate name must match with the name in the Vault server.
Certificate	Copy the certificate in PEM format.
Private Key	Enter the private key in PEM format.
Teams	Allows access of the certificate to specific teams.

   e. Save your changes.

      The certificate is added to the **Certificates** page.

Results

You have added certificate to the Deploy secret store.

What to do next

You can test the integration. See Testing Vault certificate integration on page 2212.

Vault integration test

You can test the Vault integration to ensure that HCL DevOps Deploy (Deploy) can retrieve values from the Vault server.

After you add Vault AppRole or LDAP users to the Deploy secret store, you can test whether the integration is successful. You can test the Vault integration before you use the Vault property in a plug-in step to ensure the step doesn't fail.

Testing Vault AppRole integration

You can test the Vault AppRole integration before you use the Vault property in a plug-in step.

Before you begin

You must have created a secret store and added an AppRole to the HCL DevOps Deploy (Deploy) secret store. See Adding Vault AppRoles to the Deploy secret store on page 2206.

1. On the web UI, click **Secret Stores > *vaultname* > App Roles**.
2. Click the menu options on the right-side of the desired AppRole.
3. Click **Test**.
   The **Test Vault AppRole** window is displayed.
4. Enter the Vault key or property you want to test in the **Property** field.
5. If the secret ID is not configured for the role, specify the secret ID of the AppRole in the **Secret Id** field.
6. Click **Test**.

   A message is displayed which indicates whether the test is successful or failed.

Results

You have tested the Vault AppRole integration.

What to do next

You can use the secret store property in a plug-in step. See Secret store property usage on page 2212.

Testing Vault LDAP integration

You can test the Vault LDAP integration before you use the Vault property in a plug-in step.

Before you begin

You must have created a secret store and added an LDAP user to the HCL DevOps Deploy (Deploy) secret store. See Adding Vault LDAP users to the Deploy secret store on page 2208.

1. On the web UI, click **Secret Stores > *vaultname* > Ldap Users**.
2. Click the menu options on the right-side of the desired LDAP user.
3. Click **Test**.
   The **Test Vault Ldap User** window is displayed.
4. Enter the Vault key or property you want to test in the **Property** field.
5. If the password is not configured for the user, then specify the password of the LDAP user in the **Password** field.
6. Click **Test**.

   A message is displayed which indicates whether the test is successful or failed.

Results

You have tested the Vault LDAP integration.

What to do next

You can use the secret store property in a plug-in step. See Secret store property usage on page 2212.

Testing Vault certificate integration

You can test the Vault certificate integration before you use the Vault property in a plug-in step.

Before you begin

You must have added a certificate to the Deploy secret store. See Adding Vault certificate to the Deploy secret store on page 2209.

1. On the Web UI, click **Secret Stores > *vaultname* > Certificates**.
2. Click the menu options on the right-side of the desired certificate.
3. Click **Test**.
   The **Test Vault Certificate** window is displayed.
4. Enter the Vault property you want to test in the **Property** field.
5. Click **Test**.

   A message is displayed which indicates whether the test is successful or failed.

Results

You have tested the Vault certificate integration.

What to do next

You can use the secret store property in a plug-in step. See Secret store property usage on page 2212.

Secret store property usage

At times, you may need to retrieve values from Vault and add them to a plug-in step. For example, `start tomcat` plug-in step from the Apache Tomcat plug-in. After you create a HCL DevOps Deploy (Deploy) secret store and add users to it, you can add an input property for Vault at any of the levels where secure passwords are allowed. For example, at application level or at resource level. You can use the Vault property in an automation or source configuration plug-in step.

You can retrieve user credentials from the Vault by using the following property in a plug-in step:

```
${p:secret:vault:<secretStoreName>:<vaultUsername>:<vaultSecretPath>:<vault-key>}
```

Here,

- *<secretStoreName>* is the name of the Deploy secret store.
- *<vaultUsername>* is the Vault AppRole name or LDAP user name whose credentials are required to run the plug-in step.
- *<vaultSecretPath>* is the path to the Vault secret.
- *<vault-key>* is the Vault key or property.

When the process runs, on reaching the step where the property is defined, the process validates and authenticates the credentials from the Vault.

Security teams

Users and groups on the server are assigned roles when they are added to teams.

Teams provide two important functions. First, a team provides the mechanism to assign roles to users. When a user is assigned a role, all permissions that are granted to the role are automatically granted to the user, but only for objects that the team has access to. Second, teams secure objects such as applications, components, and environments. When a team is attached to an object, only team members with the appropriate permissions can interact with the affected resource.

The System Team that comes with HCL DevOps Deploy (Deploy) has all permissions granted for all Standard security types for its Administrator role.

Creating a team

You can create a team by adding members to manage roles and access permissins for the HCL DevOps Deploy (Deploy).

Before you begin

You must have permissions to create and edit teams.

About this task

You can use the **Teams** tab to create a team. Additionally, you can view and manage the existing teams by adding members.

1. Log in to Deploy.
2. Click **Settings > Teams**.
   The **Teams** page is displayed.
3. Click **Create Team**.
   The **Create Team** dialog is displayed.
4. Type a name for the team in the **Name** field.
5. Optional: Type a description for the team in the **Description** field.
6. Click **Save**.

Results

You have created a team.

What to do next

You can assign users to the team. See, Assigning users to teams on page 2214.

Related information

Mapping teams to objects on page 2215

Roles and permissions on page 2187

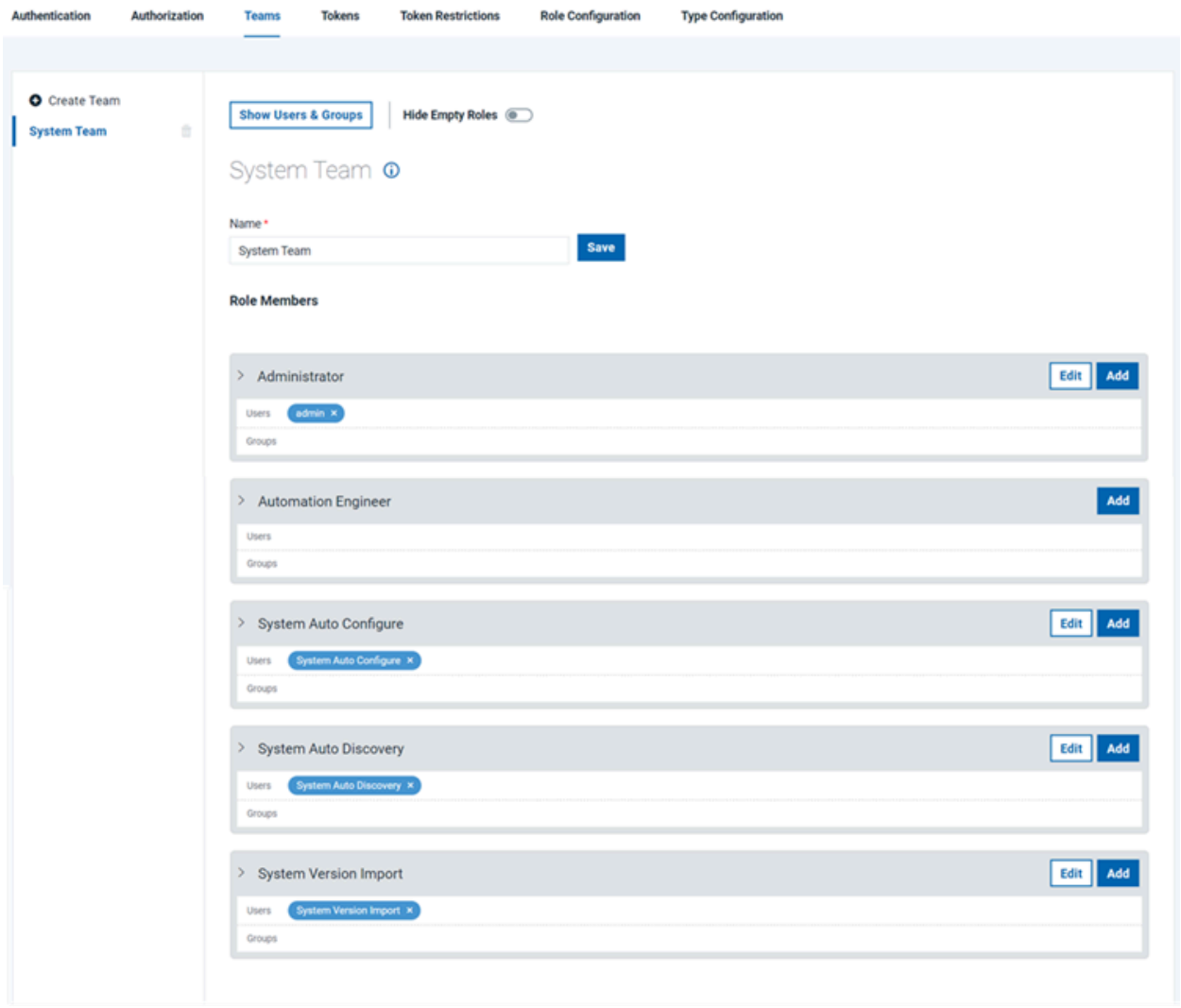
Assigning users to teams

Assign users to teams so that users can be assigned to roles and work with objects that are associated with the teams.

1. On the server, click **Settings > Teams**.
2. In the **Teams** pane, select the team to add users to.

   Example

   When a team is selected, all previously defined roles are available. In the following figure, the **System Team** is selected. All roles are displayed, and users are listed below their assigned roles. Users and groups are added to a team by assigning them to a specific role or roles.



3. To add users to roles, do one of the following actions:

   Choose from:

   - Click **Add** for the target role, and then select the appropriate check boxes from the list of users and groups.
   - Click **Add Users and Groups**, and then drag users and groups onto target roles.

What to do next

Assign teams to objects, such as applications, environments, components.

Mapping teams to objects

In addition to assigning a team to an object (such as a component or application), when you create the object, you can use the **Teams** page on the server to map teams to objects.

Before you begin

To map a team to an object, you must:

- be a member of the team
- have the create and/or manage teams subpermission enabled for that type of object

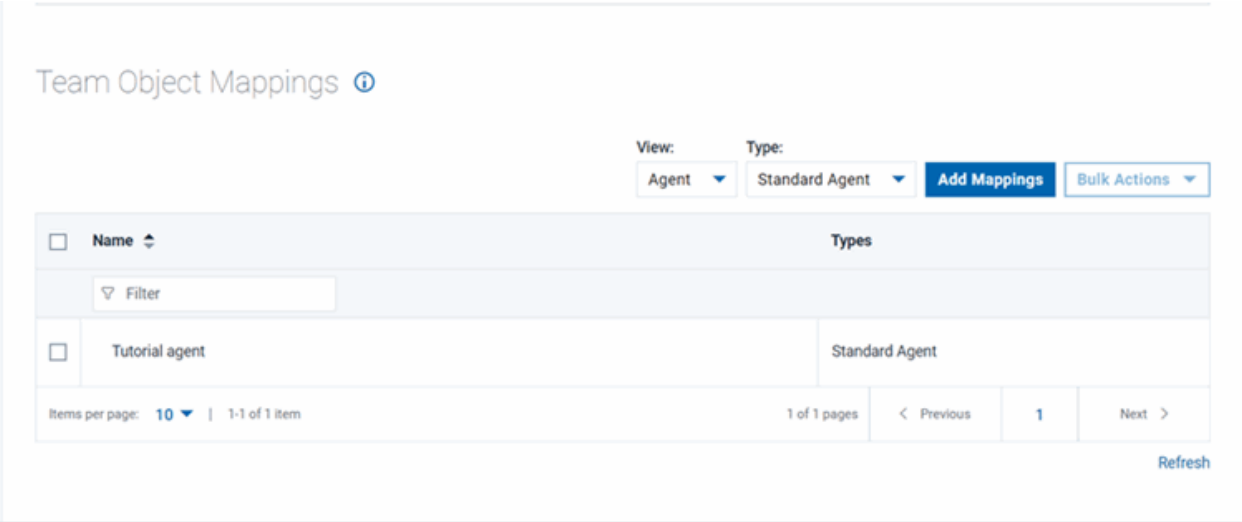
About this task

You can assign teams to many of the objects that you create. When you create an object, you can assign teams by using the associated **Create item** window. You can also assign teams by using the **Configuration** page for the object. You can also map teams to objects as described here. When you assign a team to an object, such as an environment, only team members with the appropriate permissions can interact with the affected object.

Note:

- By default, new objects are assigned to all teams that you are a member of. To specify which teams to assign new objects to, see Managing the teams and users for your account on page 2094.
- Starting with 7.3.2, if a user belongs to multiple teams, the role permissions from the most restrictive team determines the user's overall access.

1. On the server, click **Settings > Teams**.
2. From the list of teams, select the team to use.
3. Under **Team Object Mappings**, in the **View** list, select the type of object.

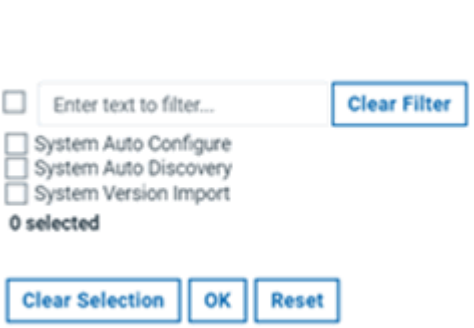


Result

The table shows the objects that are associated with the team.

4. From the **Type** list, select the security type.
5. Click **Add**.

   A list of user-created objects that are not already assigned to the current product area is displayed.



6. Select the appropriate check boxes for the objects to assign to the team, and then click **OK**.

Results

Teams that are mapped to objects are displayed on the associated object page.

Assigning teams to objects

You can assign a team to an object (such as a component or an application) when you create the object.

About this task

When a team is assigned to an object, only team members with the appropriate permissions can interact with that object. You can assign your teams to the objects that you create. To assign a team, you must be a team member and have the manage teams permission.

You can assign teams to an object when you create an object in the associated **Create item** window or by using the **Configuration** page after the object is created.

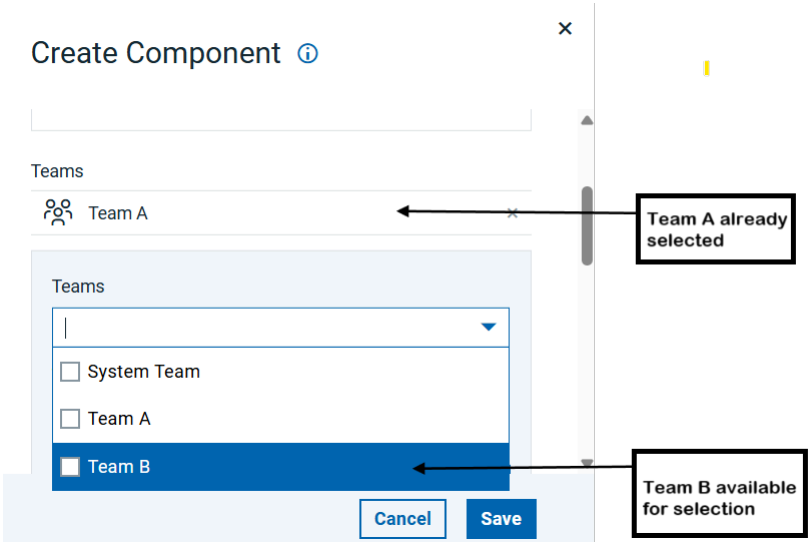
You can also assign teams by mapping teams to objects by using the mapping security feature. See Mapping teams to objects on page 2215.

Note:

- By default, new objects are assigned to all teams that you are a member of. To specify which teams to assign new objects to, see Managing the teams and users for your account on page 2094.
- Starting with 7.3.2, if a user belongs to multiple teams, the role permissions from the most restrictive team determines the user's overall access.

With the **Create with Team Assignment** permission enabled, you can add any additional team and type combinations to an object if you have the **Create** permission on at least one team-type mapping. With this permission, you can also add those team/type combinations in which you are not even a member.

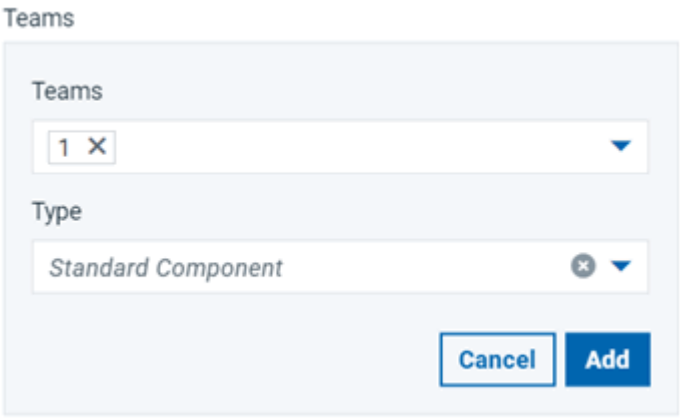
For example, there are two teams: Team A and Team B. Team A is mapped with a standard component type. Team B is mapped with a standard component and component type 2. You are a member of Team A only and have the **Create with Team Assignment** permission enabled. As a member of Team A with **Create with Team Assignment** permission, when creating a component in HCL DevOps Deploy (Deploy), Team A is automatically selected. You can add Team B in the **Add Teams** field, even if you are not a member of Team B, provided both teams are mapped with standard component type. When editing a component, you cannot unmap Team A since you are a member, but you can select Team B as both teams share a standard component type and you are a member of one.



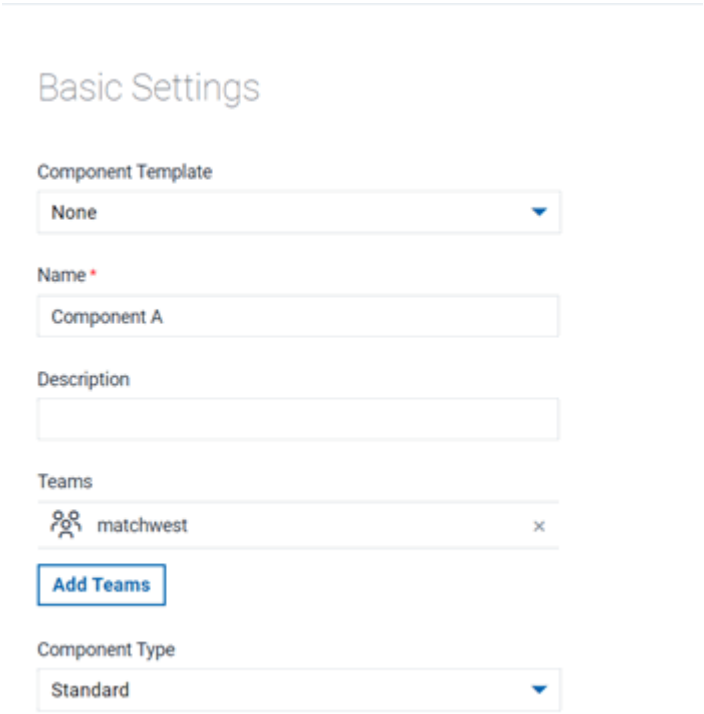
To assign teams to objects:

1. On the server, open the **Configuration** pane or the **Create** dialog box for an object, such as a component. For a list of product areas that you can assign teams to, see Security types on page 2219.
   Example

In the following figure, the **Teams** field opens in the **Create Component** window.



2. Click the checkbox to select a team from the **Team** list.
3. Select a security type for the object from the **Type** list.
   The type defines the set of object permissions that are available to the team. Security types for the current object type are available. Initially, a type of object, such as components, has a single type that is defined for it, the standard type. For information about security types, see Security types on page 2219.
4. Click **Add**.
   More than one team can be added to an item.
   Result
   The selected team or teams can interact with the object immediately. Teams and their security type are listed on the **Configuration** page that is associated with the object. In the following figure, the matchwest team with the cHolder security type is assigned to a component.



5. **Optional:** To remove a team from an object, click the cancel icon ✕ for the team.

Security types

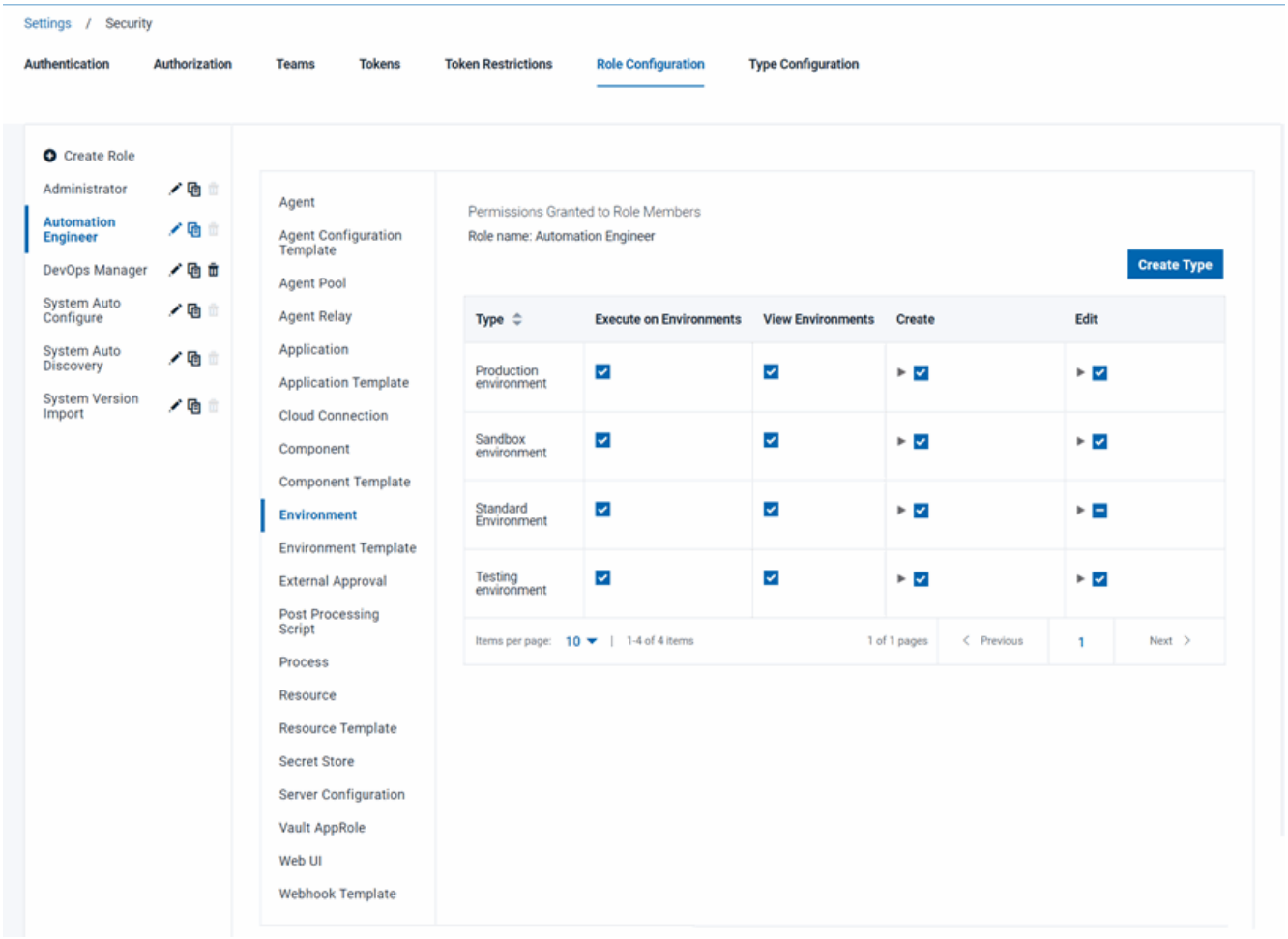
Security types define categories of objects on the server. You can put all objects into a single category, or you can put them in separate categories and give roles different access rights to different categories.

By default, all object types on the server have a standard security type. For example, applications have the Standard Application security type, and environments have the Standard Environment security type. You do not have to create more security types unless you want to give different levels of access to different roles.

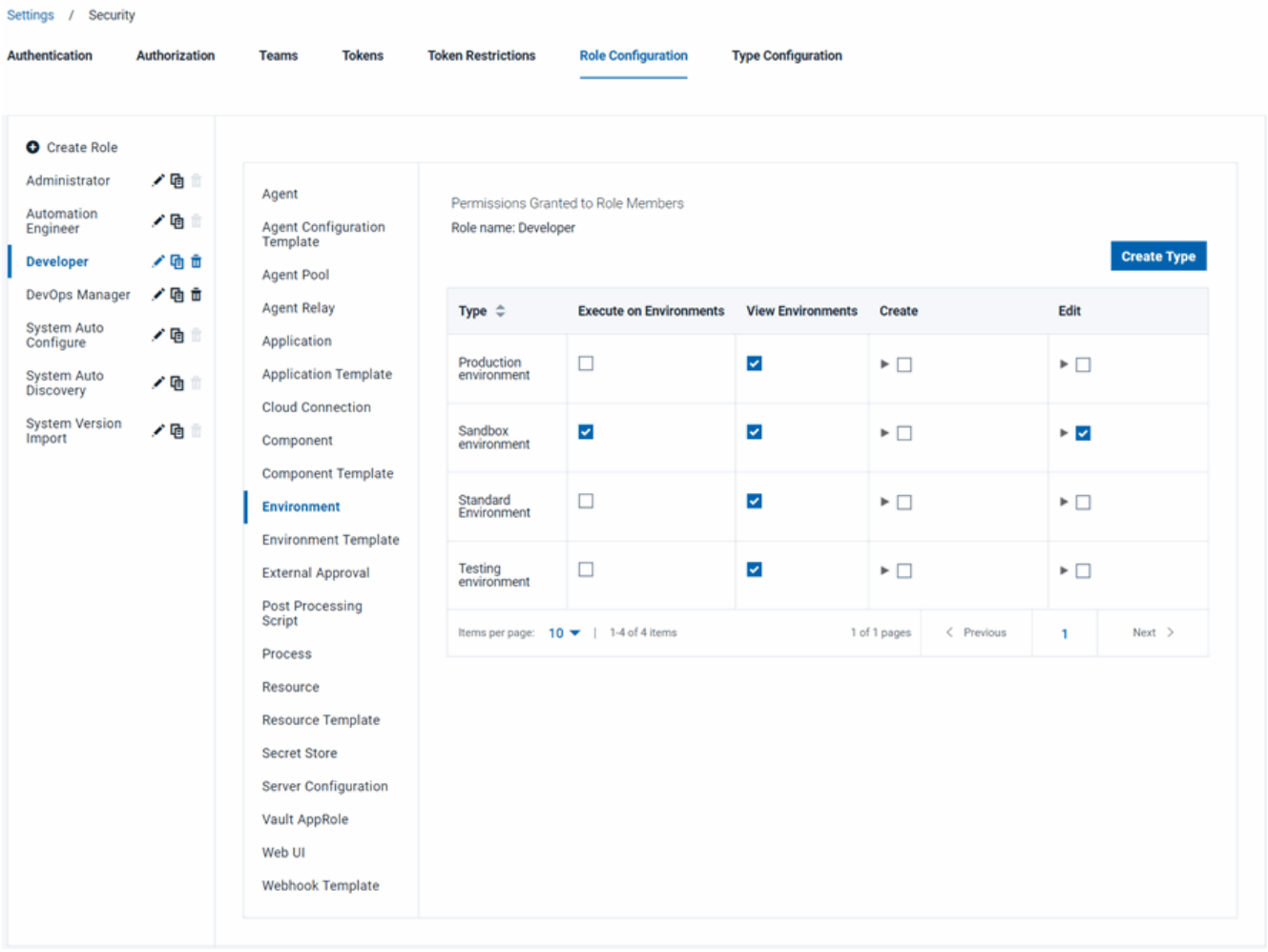
For example, suppose that you have different environment categories for different purposes, such as production environments, testing environments, and sandbox environments. You want to give different users different levels of access to these environments. You can allow only development superusers to modify production environments but allow other users to view the state of these environments. You can also allow anyone to modify sandbox environments.

In this case, you create security types for different categories of environments and assign the environments to those types. Then, you create roles for different levels of access to the environments. For example, you might create a deployment superuser role that can create and edit environments. Then, within that role, you specify the access that users have for each type of environment.

For example, the following figure shows a role that is named `Development superuser`. This role has full access to production environments, testing environments, sandbox environments, and standard environments.



A role for less-privileged users might be able to view and edit existing sandbox environments but only view production environments. The following figure shows the permissions for the `Developer` role. This role can view and edit existing sandbox environments. It can view but not edit production environments.



The System Team has access to everything on the server according to the permissions in the standard security type. For example, the members of the System Team have the environment permissions that are listed in the Standard Environment security type for all environments.

For information about assigning permissions to roles, see Creating roles and assigning permissions on page 2188.

Creating security types

Create security subtypes to help you manage access to objects through the server.

About this task

Except for the web UI and server configuration product areas, you can create security subtypes for most object types, such as applications, environments, and components.

1. On the server, click **Settings > Type Configuration**.
2. From the list of object types, select the type to add a security subtype to.
3. Click **Create Type**.

4. In the **Create Type** dialog box, enter a name for the type.
5. Click **Save**.

What to do next

Assign security subtypes to roles and grant permissions to the roles. Agent subtypes that are assigned to roles, for example, can have view and edit permissions that are granted or withheld. See Creating roles and assigning permissions on page 2188.

To delete a user-created security type, on the **Type Configuration** page, select the object type, and then click **Delete** for the type from the row options. Standard types cannot be deleted.

Tokens

Tokens provide authorization for agents, agent relays, users, and external systems or applications from the server. Agents use tokens when they run process steps and communicate with the HCL DevOps Deploy (Deploy) server and external services.

Users can use tokens with the command-line interface (CLI) client instead of supplying a user name and password in certain situations. For information, see Command-line client (CLI) reference on page 2591.

Follow these steps to create a token. You can also create a token through the CLI; see createAuthToken on page 2644.

1. Log in as a user with the **Manage Security** permission. See Setting server configuration security on page 2190.
2. On the server, click **Settings > Tokens > Create Token**. The **Create Token** dialog box appears.
3. From the **User** list, select the user for the token. You can limit the actions available to the token by applying a token restriction to it. For information about creating and applying token restrictions, see Restricting authentication tokens on page 2223.

   Note:

   If you are using the token to integrate with IBM DevOps Release, you must specify the administrator.

   If the token is for an agent relay that is intended to use component version replication, the user must have a role that has the Read Artifact Set List permission. For information about the Read Artifact Set List permission, see Setting server configuration security on page 2190.

4. Specify the expiration date and time.
5. To limit the use of the token to certain IP addresses, specify one or more IPv4 addresses in CIDR notation in the **Allowed IPs** field, such as `10.15.10.0/24`.

Note: If you are using clustered servers as described in Setting up high-availability clusters on page 2062, leave this field blank.

6. To map token restrictions to user tokens within the Deploy server during token creation, scroll through the created restrictions in the **Auth Token Restriction** field and select the desired list. Select **New** and fill out the required fields to create a new Auth Token Restriction. This feature allows you to restrict a token's usage down to specific REST API endpoints. For example, locking down an administrator user's token to only be able to call /cli/agentCLI/info.
7. Click **Save**.

Important: The token is shown only one time after you create it. Copy the token immediately, because you cannot see it again.

Tokens can be used immediately after they are created. You can revoke a token at any time by clicking **Delete** in the **Actions** column.

Note: How to generate AuthTokens

Authentication tokens (AuthTokens) are randomly generated as 128-bit values through the SHA-256 algorithm. These tokens are associated with a user ID and a set of restrictions when they are created. The value is produced by using a type 4 Universally Unique Identifier (UUID) and the Java Virtual Machine's (JVM) Secure Random generator. The generator provides 122 bits of entropy due to the inclusion of 6 bits of UUID variant information.

The UUID generation is managed by the JVM, which uses its configured secure random data source, which is typically `/dev/random` or `/dev/urandom`. However, this source can vary based on the JVM configuration.

The 128 bit entropy is generated by the *java.util.UUID.randomUUID()* variant, which calls the java.security.SecureRandom.nextBytes() bytes to produce 16 bytes (128 bits) of random data. The SecureRandom instance is created by a plain call of a new SecureRandom() object. You must consult the JVM documentation for details about entropy this process is not implemented in Deploy.

For complete information about the JVM, see the following resources:

- SecureRandom
- UUID

Restricting authentication tokens

Limit authentication token actions by applying token restrictions to process plug-in steps.

Before you begin

To create token restrictions, you must be in a role that has the Manage Auth Token Restrictions permission. See, Setting server configuration security on page 2190.

About this task

The HCL DevOps Deploy (Deploy) server generates an authentication token whenever it runs a plug-in process step. The token authenticates the user that is running the step and has the same permissions as that user. You can limit the actions available to tokens by applying token restrictions to them. A token restriction is a set of one or more REST API commands. When a token-restricted plug-in step runs, it uses the commands that are defined by the token restriction regardless of the user's permissions.

You apply token restrictions to plug-in steps in processes. You can apply a different token restriction to every plug-in step in a process. If you do not apply a restriction to a step, the default token restriction is used. The default token enables all the supported methods for all REST resources. You can modify the default token restriction but you cannot delete it.

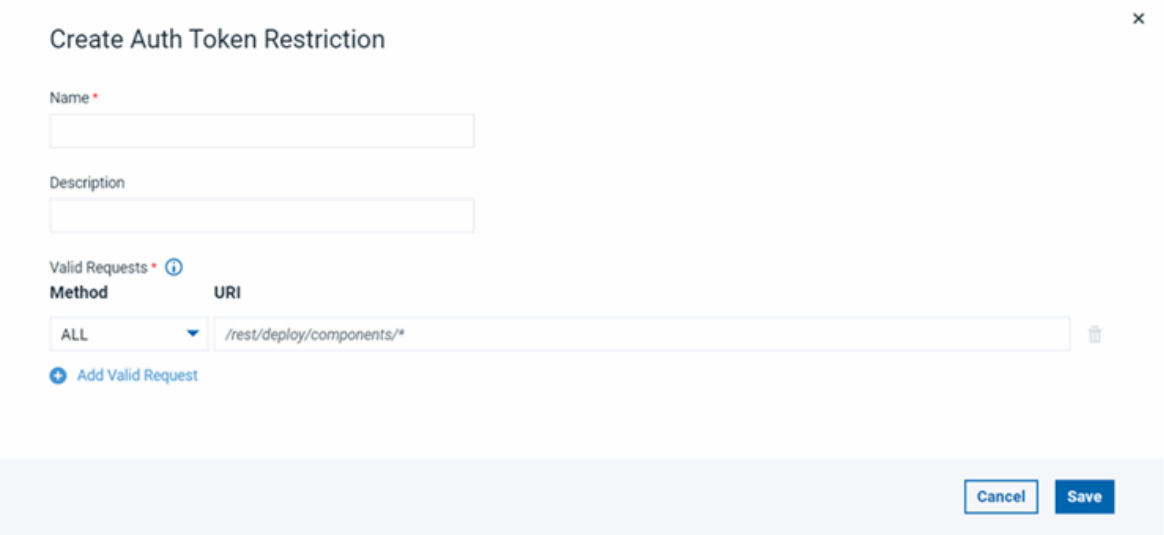
Each token restriction command consists of a verb and a URI. Verbs are methods that you can use on the REST resources. The URI represents a REST resource, such as `https://hostname:port/rest/deploy/component/`. You can define 30 commands in a single token restriction.

The methods that you can use in a command are described in the following table.

Table 713. REST API commands

Method	Description
ALL	Enable all other verbs for the associated URI.
Delete	Delete the object at the REST endpoint.
POST	Create an object of the type that is defined by the URI, or update the object at the URI.
GET	Retrieve information about the service at the URI.
PUT	Update the object at the URI.

1. On the server, click **Settings > Token Restrictions**.
2. On the **Token Restrictions** pane, click **Create Auth Token Restriction**.
3. In the **Create Auth Token Restrictions** dialog box, in the **Name** field, enter a name for the token restriction.
4. In the **Description** field, enter a description for the restriction.
5. Define one or more requests by completing the following steps:
    a. In the **Method** list, select the request verb.
    b. In the **URI** field, paste or enter the URI. The URI must start with the forward-slash character "/". You can use the asterisk (*) wildcard as the last character in the URI. For example, the following request allows information to be retrieved from all REST services:



c. If you want to define another request, click **Add**; otherwise, click **Save**.

Result

After you save the token restriction, you can copy, edit, and delete it.

Note: You cannot delete the default restriction.

What to do next

When you configure a plug-in step, you can apply a token restriction to the step. You can also create a token restriction from the plug-in process step. By default, the default token restriction is applied to each plug-in step.

Token expiration settings

Tokens provide authorization for agents, agent relays, users, and external systems or applications from the server. Agents use tokens when they run process steps and communicate with the HCL DevOps Deploy (Deploy) server and external services. In the **Security** settings area, you can set time limits for four tokens types. Two tokens types work with plug-ins that enable discovering and applying configurations. The third type is used when component versions are automatically imported. You can set the how much time elapses before the token expires and if you want to use the default auth token restrictions.

The tokens are generated automatically when the steps are run. The first three tokens use the built-in UC Auto Configure, UC Auto Discovery, and UC Version Import users and their permissions. The plug-in token has the permissions of the user who requests the process that plug-in runs from. These tokens are the same as those that you can create in the user interface and a command-line interface. See Tokens on page 2222.

Go to the HCL Plug-ins to access plug-ins that work with Deploy and provide steps that create these tokens.

The tokens are generated automatically when the steps are run. The first three tokens use the built-in UC Auto Configure, UC Auto Discovery, and UC Version Import users and their permissions. The plug-in token has the permissions of the user who requests the process that plug-in runs from. These tokens are the same as those that you can create in the user interface

and a command-line interface. See Tokens on page 2222. Go to the HCL Plug-ins website to access plug-ins that work with Deploy and provide steps that create these tokens.

To access the **Security** settings area, click **Settings**, click **System Settings**, and then scroll down.

You can specify these expiration delays for tokens:

Discovery Auth Token Expiration Delay (seconds)

Set the expiration delay in seconds for authorization tokens that are created for autodiscovery steps. You must specify at least 300 seconds. Typically, discoveries do not require more than 300 seconds (5 minutes).

Configure Auth Token Expiration Delay (seconds)

Set the expiration delay in seconds for authorization tokens that are created for auto configuration steps. You must specify at least 300 seconds. Be sure to specify sufficient time for auto configuration, because the step processes stop when the token expires.

Version Import Auth Token Expiration Delay (seconds)

Set the expiration delay in seconds for authorization tokens that are created for version import steps. You must specify at least 300 seconds. The time required to import versions depends on component size.

Plugin Step Auth Token Expiration Delay (seconds)

Set the expiration delay in seconds for authorization tokens that are created for plug-in steps. You must specify at least 300 seconds. The time required to import versions depends on component size.

Updating the agent relay authentication token password

The authentication token password for the agent relay is specified during installation of the agent relay. You can update the authentication token and provide a password for the token by updating the agent relay properties file.

Before you begin

The new authentication token for the server must be available before changing the password. For information about creating a token, see Tokens on page 2222.

1. Open the `agentrelay.properties` file located in *`agent_relay_home`*`/conf` directory.
2. Change the value for the **agentrelay.codestation.server_password** property to the new password value.
   The following is from the sample `agentrelay.properties` file. The password shown is encrypted.

   ```
   agentrelay.codestation.server_password=pbe{UlfTbYJYm2i7VIN4D0/8dYeqI6T54heliAURNECl0p4l8/3h72Fv2n3G3yylWwdCnRY57wXcLT8=}
   ```

   Specify the new password. The new password is encrypted when the agent relay is started.

   ```
   agentrelay.codestation.server_password=new_password
   ```
3. Save your changes and close the file.
4. Stop the agent relay.
5. Start agent relay .
6. Verify that the agents connected through the changed agent relay are connecting.

FIPS mode enablement

You can enable Federal Information Processing Standards (FIPS) mode on the systems that run the HCL DevOps Deploy (Deploy) server, agent, and relays to make them FIPS compliant.

While installing the server, agent, and relay with the `-fips-semeru` flag, the installer adds the FIPS properties to the system settings and the Deploy server, agent, or relay Java configuration automatically. However, you can enable the FIPS mode by adding the following FIPS properties manually:

- `-Dsemeru.fips=true` to activate the FIPS.
- `-Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced` to enable the weak enforcement policy that:
    - Limits the TLS ciphers to the approved set.
    - Uses the approved cipher implementations when provided.

      The policy does not restrict the use of non-approved algorithms for other uses, such as keystores.

Prerequisites for enabling FIPS

Ensure that you have:

- Installed a minimum supported version of IBM Semeru. See Operating systems for a list of supported versions.
- Set the JAVA_HOME on the system settings.

Enabling FIPS mode on server

You can enable the FIPS mode on your HCL DevOps Deploy (Deploy) server to make it FIPS compliant.

1. Stop the server.
2. Open the `<serverInstall>/bin/set_env.cmd` file.
3. Add the following lines to the JAVA_OPTS line:

   ```
   -Dsemeru.fips=true
   -Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced
   ```

4. Start the server.
   The following message indicates that the server has started with the FIPS mode:

   ```
   Java FIPS Support: active
   ```

Results

The FIPS mode is enabled on the server.

Enabling FIPS mode on compact agents

You can enable the FIPS mode on the HCL DevOps Deploy (Deploy) compact agent to make it FIPS compliant.

1. Stop the agent.
2. Open the `<DEPLOY_AGENT_HOME>/conf/agent/javaopts.conf` file.
3. Add the following lines to the `javaopts.conf` file:

```
-Dsemeru.fips=true
-Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced
```

4. Start the agent.

Results

The FIPS mode is enabled on the compact agent.

Enabling FIPS mode on non-compact agents

You can enable the FIPS mode on the non-compact agent to make it FIPS compliant.

1. Stop the agent.
2. Open the `<agentInstall>/bin/worker-args.conf` file.
3. Add the following lines to the `worker-args.conf` file:

```
-Dsemeru.fips=true
-Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced
```

4. Start the agent.

Results

The FIPS mode is enabled on the non-compact agent.

Enabling FIPS mode on agent relays

You can enable the FIPS mode on the HCL DevOps Deploy (Deploy) agent relay to make it FIPS compliant.

1. Stop the relay.
2. Open the `<relayInstall>/bin/worker-args.conf` file.
3. Add the following lines to the `worker-args.conf` file:

```
-Dsemeru.fips=true
-Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced
```

4. Edit the `conf/agentrelay.properties` file and add the following line:

```
fips.flag=-Dsemeru.fips=true\n-Dsemeru.customprofile=OpenJCEPlusFIPS.FIPS140-3-Weakly-Enforced
```

Note: The `\n` separating the two properties is important.

5. Start the relay.

Results

The FIPS mode is enabled on the agent relay.

toc = no

Setting your profile

As a HCL DevOps Deploy (Deploy) user, you can manage teams and roles for your account, and preferences such as display options, language for the server, and how the server shows times and dates.

Managing the teams and users for your account

You can view and manage the teams and roles that you are assigned on the server. Your permissions determine which changes you can make.

Setting the teams for your account

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > Team Manager**.
2. Manage the teams that you are a member of.
   For more information, see Security teams on page 2213.

Setting the users for your account

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > User Manager**.
2. Manage the user that you are a member of.
   For more information, see Security teams on page 2213.

Setting the user preferences

You can set the preferences such as language for the server, and how dates and times are shown on the server.

1. At the upper-right corner of the server window, click your user name, and then click **My Profile > Preferences**.
2. Choose a language for the user interface in the **Locale** field.
3. Select a format for dates in the user interface in the **Date Format** field.
4. Select a format for times in the **Time Format** field.
5. Specify the calendar type you want to use for the server in the **Calendar** field.
6. Specify which teams to assign for new objects in the **Default Teams for New Objects** list.
   By default, **System Default Policy** is selected and uses the **Default Teams for New Objects** settings defined in the system settings.
   - **All Available Teams** to add new objects to all of the teams that you are a member of.
   - **Specific Teams** to specify specific teams, and then select one or more teams.
   - **None** to specify that no new objects are added to your teams.
7. Click **Save**.

Upgrading high-availability installations

All servers that are configured for the HA feature must be on the same version. Thus, all servers must be upgraded at the same time.

Before you begin

Note: During the upgrade of HCL DevOps Deploy (Deploy), the contents of the appdata/patches directory (including all .jar files) will be appended with `.off`, preventing the existing test fixes from being loaded once the Deploy server is started post-installation. All UI test fixes applied to the Tomcat directory will be wiped by the upgrade and will not be recoverable.

If a text fix received for a reported defect is not fixed in the upgraded version, you may need to open a Support Case to request a new build.

1. Back up the database and file system.
   See Backup and recover on page 2057.
2. Ensure that no application and generic processes are running.
   Open the **Dashboard** and confirm that no processes are running. If processes are running, you can wait for them to complete or by clicking **Cancel** in the same row as each process, cancel them. You can use the Message of the Day feature (**Settings > System > General Settings**) to notify users of the planned upgrade, and request that they do not start new processes.
3. Download and extract the installation package.
   Upgrades are done with the same package used to perform new installations.
4. Shut down the servers.

   Warning: Ensure that all servers are stopped. If any servers are accessing the database when the upgrade happens, the upgrade will cause errors.

5. Ensure that each server is stopped and unable to connect to the load balancer.

   a. Check to ensure that there is no file `server.pid` in the *`server_install`*`/var` folder.

   b. Check for running system processes.

   Example

   For example, on Linux™, you can use the netstat or ps commands. For example, if the server is running on port 8443, use the command `netstat -nap | grep 8443`. You can also run `ps -ef | grep server` and look for the server system process in the results.

   c. If any server processes are still running, stop them.

   If a server process is still running, when you install the new version, you see an error that says "A previously installed version of Deploy is running. Please shutdown the running Deploy and start the installation again."
6. Run the server installation script.
7. From the command line, run the server installer program.

   Choose from:

   - On Windows™, run the `install-server.bat` file.
   - On Linux™, run `./install-server.sh`.

- To install a FIPS-compliant server, add the `-fips-semeru` parameter to the command, such as `install-server.sh -fips-semeru`.

  Note:
  - If you install a FIPS-compliant server, cloud provisioning is not available.
  - Deploy supports IBM Semeru versions 17.0.12, 21.0.4, 25.0.0 or higher.

  Depending on your system settings, you might need to run this file as an administrator.

8. When prompted for the location of the installation directory, enter the path to one of the servers.
   By specifying an existing installation, you are asked if you want to upgrade the current installation. If you answer yes, the script leads you through the upgrade steps.
9. Restart the upgraded server.
   Always use service scripts to start or stop the server. See Starting and stopping the server.
10. After the first server is back online, upgrade the other servers. After you upgrade the first node, you will not be prompted to upgrade the database when you upgrade the other nodes.

Managing locks

A *lock* ensures that processes do not interfere with one another. Normally, locks are released after they are no longer needed. Sometimes, a lock is not released, and its associated process is unable to complete. The lock management feature helps you to identify and resolve abnormal lock conditions.

About this task

Follow these steps to view and resolve locks.

1. A running process with a lock, like all active processes, is shown on the **Dashboard** tab with a status of `Running`.
   If a locked process takes longer to complete than expected, you can cancel the process from the **Dashboard**, or investigate it fully with the **Settings** tab.
2. Display the **Locks** pane by clicking the **Locks** link on the **Settings** tab (**Settings > Locks**).
   Result
   The **Locks** pane displays the following information:

   Table 714. Lock fields

Field	Description
Name	The name identifies the lock. The displayed name is a concatenation of the component or application name (depending on type) + process name + resource name.
Type	Indicates whether the process that created the lock is a component or application type. Locks can be applied only to component or application processes.
Component/Application	Displays the name of the component or application that contains the lock. Clicking an item displays (depending on the type) the **Component** pane, or **Application** pane, where you can investigate the lock.

Field	Description
Resource/Environment	Displays the name of the resource or environment that contains the lock. Clicking an item displays (depending on the type) the **Resource** pane, or **Environment** pane.
Process	Displays the name of the process that contains the lock. Clicking an item displays the process in the process editor.
Actions	Lists the available actions.

3. Resolve the lock by selecting an action:

Table 715. Lock actions

Action	Description
View Request	Displays the process log for the process that contains the lock. You can use the **Actions** field on the displayed pane to see the name of the process step that is causing the lock.
Release	Releases the lock, which enables the associated process to continue processing.

If the HCL DevOps Deploy (Deploy) server and or agents go down while a locked process is running, Deploy automatically restores any interrupted processes along with any locks they might contain after service is restored.

Acquire Lock on page 2466

Release Lock on page 2483

System properties

System properties are global variables for the server.

System properties are available on the **Settings** tab: click **Settings > Properties**.

References to system properties are similar to the following example:

```
${p:system/propertyName}
```

If you create a `SUCCESS` system variable , for example, you would refer to it as in the following example:

```
echo ${p:system/SUCCESS}
```

Output in this case:

```
SUCCESS
```

System settings

System settings allows you to manage global settings for components, clean up, password requirement, security, mail-server, and much more.

Example

The following tables list the settings for several system-wide settings for the server.

Table 716. General settings

Field	Description
External Agent URL*	Specify the URL that the HCL DevOps Deploy (Deploy) server provides to agents as their point of contact. When an agent wants to contact the server, it retrieves this URL and opens a connection on it. This URL must point to the HTTP or HTTPS port of the server.
External User URL*	Specify the URL that users enter to connect to the Deploy server. The URL is also used when creating a notification template.
Message of the Day	A custom notification message that is displayed when users log in to the dashboard. This field supports standard Markdown syntax, which you can use to include formatted headings, bold text, numbered or bulleted lists, inline code blocks, and internal or external hyperlinks.
Validate Agent IP	When enabled, the agent IP and host name are validated against values that are used the first time that the agent is connected to the Deploy server.
Create Default Children for Plugins	When enabled, empty groups are created when plug-ins use the auto discovery feature.
Default Locale	Select a default language for the user interface.
Default Snapshot Lock Type	Determines the default lock type when users create snapshots. Lock types are the following values: **Only Component Versions**, **Only Configuration**, or **Component Versions and Configuration**. The default value is **Component Versions and Configuration**. Changes are effective after the server is restarted.
Require a Comment For Process Design Changes	When enabled, prompts users to provide a comment when a process design is updated.
Enable Maintenance Mode	When enabled. stops activities on the server so you can run important tasks that require the server to be idle.

Table 717. Component settings

Field	Description
Components copy to CodeStation by default	When enabled, newly created components copy artifacts to CodeStation by default. You can change this setting on individual components even when this option is enabled.
Enforce Deployed Version Integrity	When enabled, deployed component version artifacts cannot be changed.
Automatic Version Import Check Period (seconds)*	Specify the time-period (in seconds) between the times that Deploy polls components for new versions. If this value is changed, the server must be restarted before the change becomes effective. You can set this value to 15 seconds or higher.
Use Agent Tag For Integration	When enabled, allows agent that is identified by a specific tag to import component versions. If this setting is selected, the **Agent Tag for version imports** field is shown. Otherwise, the **Agent for Version Imports** field is displayed. For information about tagging objects, see Adding tags to objects on page 2521
Agent for Version Imports	Specify the agent that is used to import component versions. The agent must have access to component artifacts. The agent becomes the default agent for new components. This value can be overridden when a component is created. This field is displayed if the **Use Agent Tag for Integration** toggle button is set to Off. To import artifacts from the server location, install an agent in the same location and specify that agent.
Version Import Records to Retain	Specify the number of version import records to retain during cleanup. Default is 100.
Enable Safe Edit of Component Processes	When enabled, you can test component processes before making them available in normal environments. You can make updates to your component process, test those updates safely, and then promote to a new version when you are ready.
Require Approval to Promote	When enabled, any draft component process must be approved by an authorized user before it can be promoted and used in deployments.

Table 717. Component settings (continued)

Field	Description
Component Process	
Allow Process Locking	When enabled, users with 'Manage Process Lock' permission can lock component processes for editing. Users can also force unlock a process locked by another user.
Require Process Locking	When enabled, users will not be able to edit component processes without locking the process first.
Server Storage Display '100%' Size (MB)	Define the size in megabytes of '100%' storage in the dashboard server storage display. This setting only affects the dashboard display and does not enforce a limit for artifact uploads.

Table 718. Artifact cleanup settings

Field	Description
Daily Cleanup Start Time*	Enter the time when versions are cleaned. The value must be an integer from 0 (midnight) to 23 (11 pm). To preview the component versions that are scheduled for cleaning, click **Preview Version Cleanup**. See Cleaning up component versions on page 2136.
Default Application Snapshot Retention (days)*	Enter the number of days application snapshots are kept by default. A value of `-1` means that they are kept indefinitely. Component and environment settings can override this setting.
Default Version Retention for Component (days)*	Enter the number of component versions to keep by default. A value of `-1` specifies that all are kept. Component settings can override this setting.
Default Number of Versions to Retain for Component*	Enter the number of versions to retain for each component.
Default Version Retention for Environment (days)*	Enter the number of environment versions to keep by default. A value of `-1` specifies that all are kept. Environment settings can override this setting.
Default Number of Versions to Retain for Environment*	Enter the number of versions to retain for each environment.

Table 718. Artifact cleanup settings (continued)

Field	Description
Archive Path	Specify the path where the compressed file that contains the archived component versions is written. If this field is blank, the compressed file is not written, and no archive is kept.
Preview Version Cleanup	Click **Preview Version Cleanup** to see the versions that are archived using the current settings. See Cleaning up component versions on page 2136.

Table 719. Process history cleanup settings

Field	Description
Daily Cleanup Start Time	Specify the time that the daily cleanup process starts.
Daily Cleanup Duration (hours)	Specify the number of hours that the cleanup process runs daily.
Enable Deployment History Cleanup	When enabled, deployment history files are reduced. See Cleaning up deployment history files on page 2133.
Days to Retain Deployment History	Enter the number of days to retain deployment history. This setting can be overridden with the related environment setting.
Enable Generic Process History Cleanup	When enabled, the generic process history files are set to delete as per the **Days to Retain Generic Process History** setting.
Days to Retain Generic Process History	Enter the number of days to retain generic process history. This setting can be overridden with the related environment setting.
Enable Process Step Log Cleanup	When enabled, the process step log cleanup starts. The process step logs are deleted according to the values that you set without deleting other history records.
Days to Retain Process Step Logs	Enter the default number of days to retain process step logs for generic processes not associated with deployments, and for environments if the number of days is not specified directly in the environment settings. Valid range is 0 or greater.

Table 720. Audit Log settings

Field	Description
Enable Audit Log Entries for Read Events	This option is disabled by default on new installation. When disabled, no READ events are recorded in the audit log.
Enable Audit Log Cleanup	When enabled, audit log entries older than the number of days specified in the **Days to Retain Audit Log Entries** field are removed beginning at the time specified by the **Daily Cleanup Start Time** field. See Configuring log rotation settings on page 3002. Deleting audit logs from **Settings > Audit Log > Cleanup Log** creates a log entry of event type Audit Log Cleanup that are non-deletable.
Daily Cleanup Start Time	If **Enable Audit Log Cleanup** is enabled, specify the time the audit log cleanup starts. The default value is 12:00 AM.
Days to Retain Audit Log Entries	If **Enable Audit Log Cleanup** is enabled, specify the number of days that audit log records are retained. The default value is 30.

Table 721. Password requirement settings

Field	Description
Minimum Password Length*	Define the minimum length for passwords.
Require Uppercase Characters in Passwords	When enabled, passwords must contain at least one uppercase character.
Require Lowercase Characters in Passwords	When enabled, passwords must contain at least one lowercase character.
Require Special Characters in Passwords	When enabled, passwords must contain at least one special character.
Require Number Characters in Passwords	When enabled, passwords must contain at least one numeric character.
Require Password to be Unique From User Information	When enabled, passwords must not match username, actual name, or configured email.

Table 722. Security settings

Field	Description
Discovery Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Auto Discovery Steps to expire. The value must be 300 seconds or more.

Table 722. Security settings (continued)

Field	Description
Configure Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Auto Configure Steps to expire. The value must be 300 seconds or more.
Version Import Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for Version Import Steps to expire. The value must be 300 seconds or more.
Plugin Step Auth Token Expiration Delay(seconds)	Specify the delay in seconds for auth tokens that are created for plug-in steps to expire. The value must be 300 seconds or more.
Use Default Auth Token Restriction if not specified	When enabled, the default authentication token restriction for processes that are created in Deploy is used. For these versions, the default token is applied to the most current process version but not other versions. For more information, see Restricting authentication tokens on page 2223.
Require Token Restriction	When enabled, the None option is removed from the Auth Token Restriction drop-down when creating or editing an authentication token. This makes Auth Token Restriction a required field, but the user can still set a highly permissive token restriction.
"Remember me" duration (hours)	Specify the number of hours a user can optionally be remembered and remain logged in.
Maximum Login time (hours)	Define the maximum duration (in hours) a user can stay logged in. When set, it forcibly disconnects the remote users from server when the logon hours expires. This setting overrides the **"Remember me" duration (hours)** setting. You can enter `0` or a negative value to disable the setting.
Logout should invalidate all "Remember Me" cookies	When enabled, invalidates all "Remember Me" cookies for a user when they explicitly log out.
Default Teams for New Objects	Select **All Teams Available to User** to add all teams that are available to user by default when a new object is created. When set to None, no teams are assigned to the users when new objects are created. Users can override this setting while creating objects.

Table 722. Security settings (continued)

Field	Description
Trusted proxy list	Enter a regular expression pattern to match IP of the trusted proxy servers. This value affects all servers. You must restart the servers for the changes to reflect. If left blank, the proxy headers are not trusted from incoming connections, which further may affect auth token IP restriction validation and IP logging.
User Idle Days to Lock	Specify the number of days after the user's last login date before locking the user account. You can enter 0 to disable user account locking.
User Idle Days to Delete	Specify the number of days after the user's last login date before deleting the user account. You can enter 0 to disable user account deletion.
Preview Idle User Lock	Click **Preview Idle User Lock** to load the information about user accounts that will be locked or deleted.
Realm for Bearer Authentication	Select an authentication realm to be used for the bearer authentication.

Table 723. Mail server settings

Field	Description
Mail Server Host	Enter the host name of the mail server that is used for notifications. Deploy can send notifications to users based on user-configured trigger events (to set up notifications, see Creating Notifications in a Notification Scheme on page 2081). Deploy requires an external SMTP mail server to send messages. To disable notifications, leave the field blank.
Mail Server Port	Enter the SMTP port that is used by the notifications mail server.
Secure Mail Server Connection	When enabled, the SMTP server connection is secured.
Mail Server Sender Address	Enter the sender's email address for email notifications.
Mail Server Username	Enter the user name for sending email notifications.

Table 723. Mail server settings (continued)

Field	Description
	Some email servers and firewalls treat emails with different sender and user names suspiciously. Therefore, consider specifying the same name for both fields.
Mail Server Password	Enter the user password for sending email notifications.
Max Recipients Per Email	Specify the maximum number of recipients to include in a single email. Mutliple emails will be sent when this is exceeded.

Table 724. External Approval settings

Field	Description
Resource Type to Run External Approvals	Choose any of the following options: • None: to use no agent or agent pool to run external approval processes. • Agent: to use an agent as the default resource type to run external approvals. Specify agent in the **Agent for External Approvals** field which will be used for external approval processes. • Agent Pool: to use an agent pool to specify a group of agents to run external approvals on instead of using a specific agent. Specify agent pool in the **Agent Pool for External Approvals** field which will be used for external approval processes. For more information, see Managing external approval processes on page 2392.

Table 725. Legacy Configuration Options

Field	Description
Enable "Prompt On Use"	When enabled, the option to set Prompt On Use for properties in workflow steps is displayed.
Fail Processes With Unresolved Properties	When enabled, any running process which comes across a property which it cannot resolve will automatically fail.
Component Environment Properties Override Environment Properties	When enabled, component environment properties overrides environment properties when using `${p:propName}` notation as opposed to `${p:environment/propName}`.
Enable Deleting Resources From Environments	When enabled, allows you to delete resources that are mapped with environment while deleting that environment.

Table 725. Legacy Configuration Options (continued)

Field	Description
	Note that the resources are deleted globally and affects the other environments that uses the same resource tree.
Show Compliancies Automatically	When enabled, environment compliancies are loaded automatically on the **Application Environment** and **Snapshot** pages rather than loading on demand.

High-availability agent communication

Agents communicate with the server over HTTP or HTTPS. These modes of communication are handled differently in a high availability scenario.

HTTP or HTTPS communication

A standard load balancer handles HTTP or HTTPS server-agent communications. Authentication is run on every request; any server can run validation for any request. You configure HTTP or HTTPS server-agent communication when you configure your cluster of servers. See Setting up clusters of servers on page 2063.

In case of server failure, user-server authentication must be redone unless the 'remember me' cookie is used.

High-availability server processes

While high-availability servers complete the same processes as standalone servers, they run slightly differently and access files in run slightly different locations than individual servers do.

In a high-availability topology, all servers use the same database and a common network storage. The server log and CodeStation repository are stored in the common network storage. Most configuration information and runtime data is stored in the shared database. Each server maintains some independent configuration information, such as the connection information for the database.

Because the servers share the database, all servers run on the same interval.

Many activities run the same on high-availability servers as on a standalone server. Some activities run slightly differently:

Importing component versions (CodeStation)

In a clustered high-availability configuration, any server can start the process that imports a component version. If components are configured for automatic imports, all servers poll for component version changes, but they are limited to the polling duration in the system settings. To prevent more than one server from importing a component version or checking for new versions, the database handles server synchronization and a server acquires a database lock before importing or polling.

Notifications

Notifications are handled by the server that runs the process or starts the related event. For more information, see Creating Notifications in a Notification Scheme on page 2081.

Running processes and handling failures

While processes are running, step failures are shown in the process log as usual. Error handling is the responsibility of the process author; for example, processes can include steps that run when other steps fail, including automatic rollback steps.

When a process is requested, a request is created in the database. Servers poll the database regularly for new process requests. The first server that finds the process request runs the requested process. Due to a limitation on the server, the process can not be transferred to another server in the cluster. Therefore, if the server fails in the middle of running a process, the process fails. The other servers in the high-availability cluster run future processes, but they can not take over processes that the failed server was running.

If an agent relay stops while a connected agent is running a process, the agent uses its failover settings to connect to other servers or agent relays. If the agent exhausts its failover options, it continues trying to reconnect. If an agent relay or server comes back online, the agent continues to run the process.

Step failures do not cause agents themselves to fail. If an agent stops while it is running a process, the server assumes that the agent is still running the process and does not mark the process as failed.

Storing reusable post-processing scripts

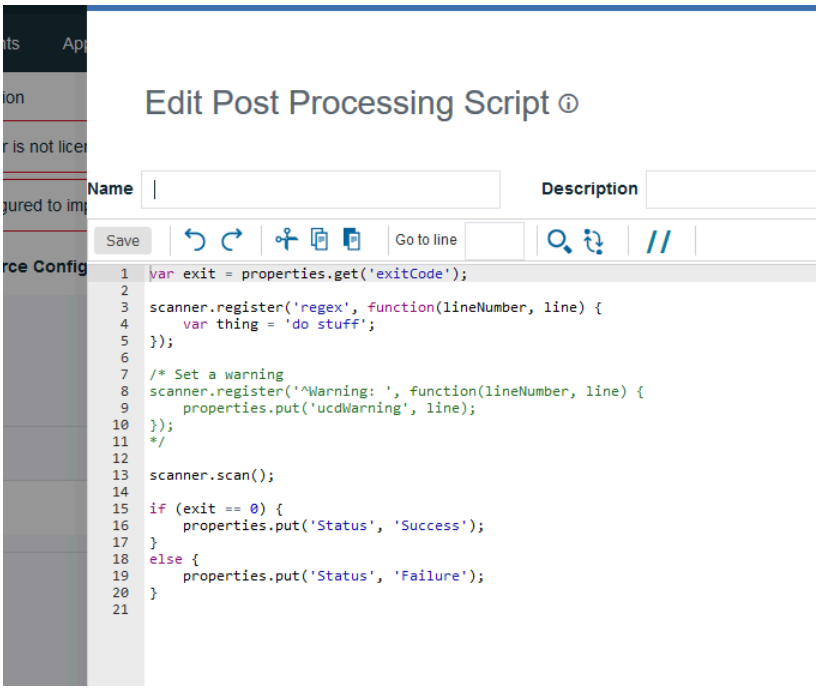
You can store post-processing scripts on the server so you can reuse them in component processes and in generic processes.

About this task

When a step finishes, the agent runs the post-processing script for that step. When the agent runs the script, it first loads the server log file. It finds the exit code property of the target step with regular expressions that are defined in the script. It then applies any actions that are defined in the script before it runs the next step.

The HCL DevOps Deploy (Deploy) server log file is normally found in the following location: *server_installation_directory*`/var/log/deployserver.out`.

1. Click **Settings > Post Processing Scripts** and then click **Create Script**.
   Result
   The **Edit Script** window opens, as shown in the following figure.



2. Specify a name for the process step.
3. Give a description for the process step.
4. Specify the teams that can access and edit the script.
5. Enter or paste the script into the **Script Body** field.
   See the roll-over help next to the field for information about the properties and variables that are available for user-defined scripts.

   Post-processing scripts must set the **Status** property to specify the status of the script. The script can specify any value for the **Status** property. For example, to specify that the script was a success, run the following command in the post-processing script:

   ```
   properties.put("Status","Success");
   ```

Creating statuses

Define component version, snapshot, and inventory statuses on the **Statuses** tab.

1. Click **Settings > Statuses**.
2. For the status type to create, click **Add Status**.
3. Configure the status in the **Add Status** window:

Parameter	Description
Name	Identifies the status and is included in many UI elements.
Description	Conveys more information about the process.
Color	Defines the status color that is displayed in the UI.

Parameter	Description
Unique	When checked, specifies that only a single instance of the status describes a component. For inventory or snapshot statuses, an application removes the status from existing versions in the environment or resource inventory. For version or snapshot statuses, the status can be used by only one version or snapshot at a time.
Required Role	Specifies the user role that is necessary for a user to assign the status.

4. Save your work when finished.
   Statuses are stored in the HCL DevOps Deploy (Deploy) database.

Tip: Default statuses are defined in an XML file, which you can edit to add your own values.

Adding version statuses

Manage component version statuses through the version options interface.

About this task

You add or remove statuses from individual component versions to provide information about the version. If you copy files to an agent without deploying them, you might apply a `Staged` status to them. When you install the version, you might set the status to `Active`.

1. To select the component, click **Components > *selected component***.
2. To select the version, click **Versions > *selected version***.
3. Click **Add a Status**.
4. Select the status from the list, and click **Save**.

What to do next

Add a status by using the Add Status to Version plug-in step, which is available from the Version plug-in.

Modeling Software Deployment

Modeling software deployment in HCL DevOps Deploy (Deploy) includes configuring components and component processes and adding those components to applications. Then, you use processes to deploy the components to environments.

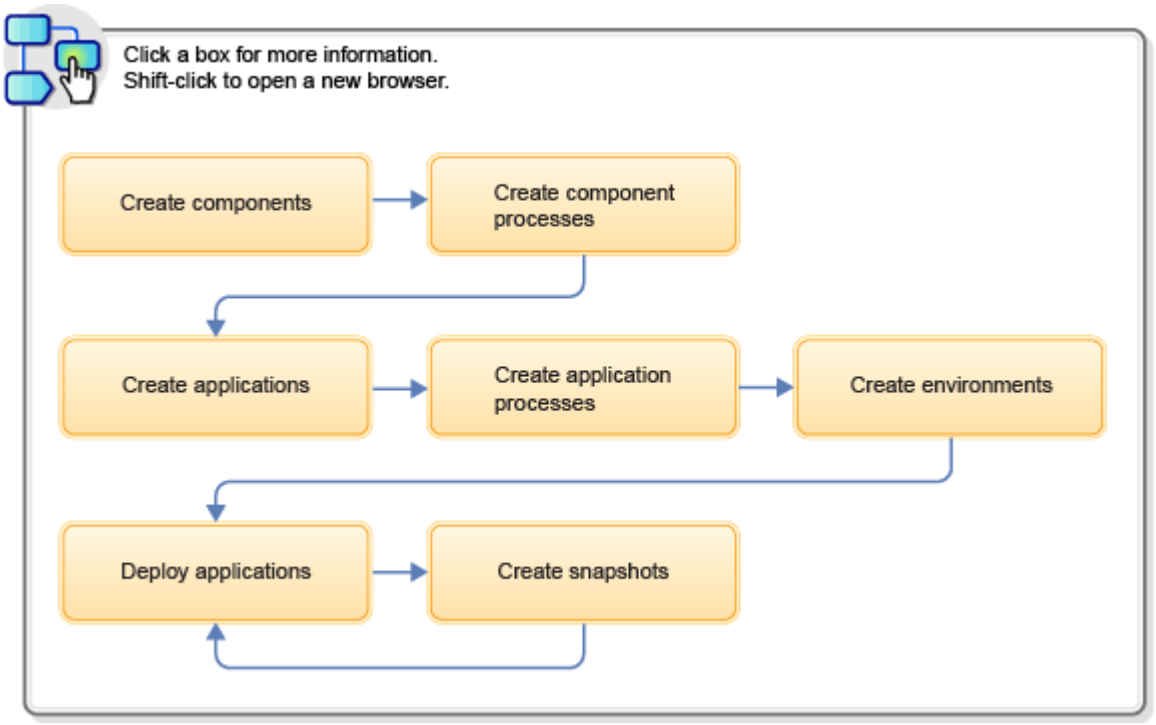
About this task

The general steps in this typical workflow include:

1. Import the parts of the application as components. See Creating components on page 2248.
2. Create processes that deploy components. See Component processes on page 2299.
3. Create an application to group the components together. See Applications on page 2344.
4. Create an application process that runs each component process. See Creating application processes on page 2429.
5. Create one or more environments on which to deploy the components. See Creating environments on page 2354.

6. Run the application process to deploy the components. See Deploying applications on page 2526.
7. Take snapshots of working deployments so you can save these arrangements and deploy them to other environments. See Creating snapshots on page 2383.

The following diagram illustrates this workflow:



1. Creating components on page 2248
2. Component processes on page 2299
3. Applications on page 2344
4. Creating application processes on page 2429
5. Creating environments on page 2354
6. Deploying applications on page 2526
7. Creating snapshots on page 2383

Dashboard

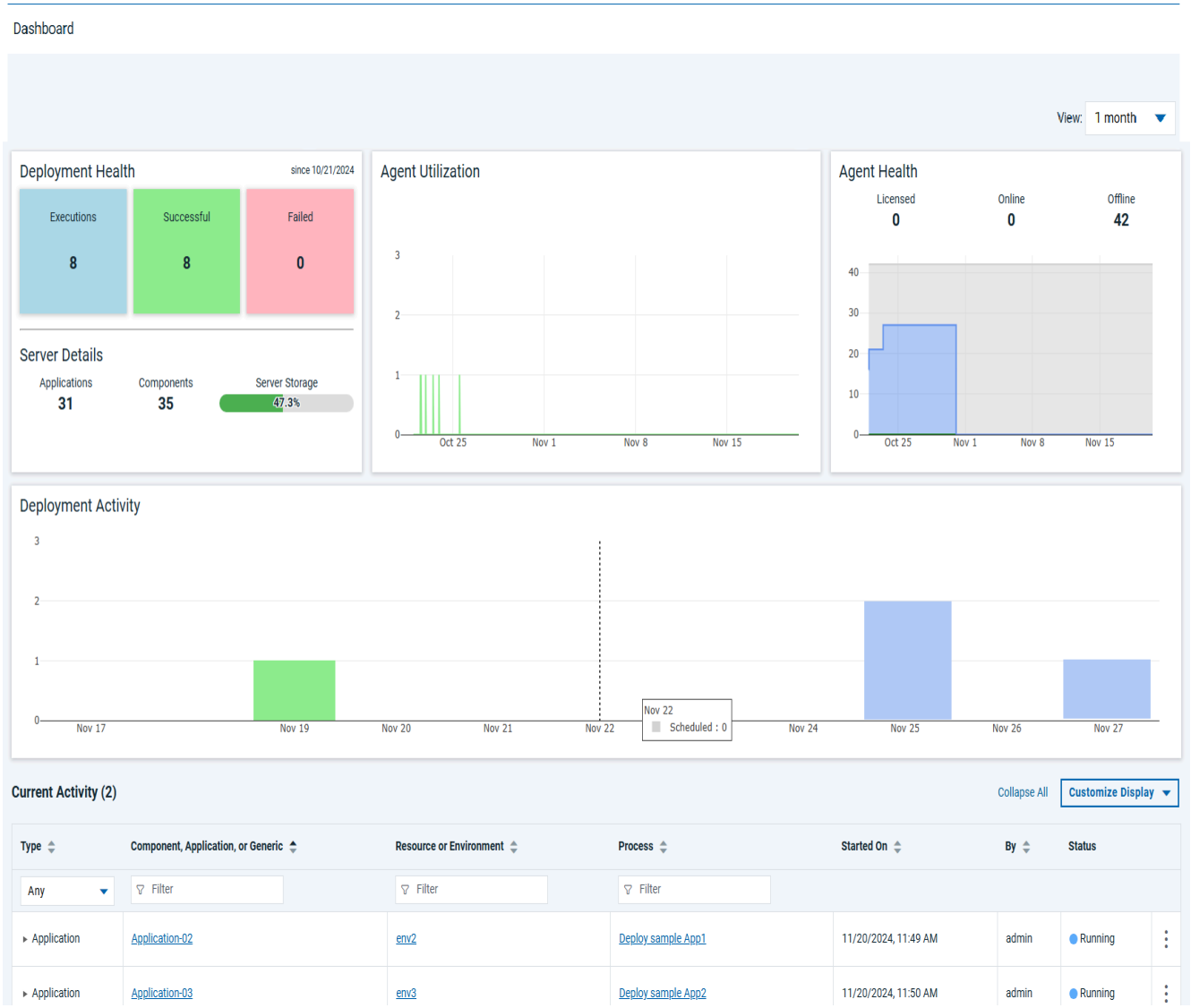
The Dashboard is a live page that gives you real-time insights into the status of your deployments and agents through graphs and charts.

The dashboard has multiple widgets that help you track, analyze, and monitor your deployments and agents. The following widgets are available in the Dashboard:

- Deployment Health on page 2246
- Agent Utilization on page 2247
- Agent Health on page 2247

- Deployment Activity on page 2247
- Current Activity on page 2247

The following screenshot is an example representation of the Dashboard page:



Note: By default, the **Deployment Health**, **Agent Utilization**, and **Agent Health** widgets display data covering the last one month. You can view a custom time frame such as the past week, three months, or one year by clicking the **View** drop-down menu in the upper-right corner of the dashboard and selecting your preferred range.

Deployment Health

The Deployment Health widget gives you an overall status of your deployments in the last month.

Widget part	Description
Executions	Displays the total number of deployments executed.
Successful	Displays the total number of successful deployment jobs.
Failed	Displays the total number of failed deployment jobs.
Server Details	• **Application**: Displays the total number of applications created in the system. • **Environment**: Displays the total number of environments created for these applications. • **Server Storage**: Displays the amount of CodeStation storage being used by artifacts in percentage. You can define the storage size to display in **Server Storage Display '100%' Size (MB)** field on the **System Settings** page.

Agent Utilization

The Agent Utilization widget is indicative of the agent usage data. It displays a time series chart along with counts to show the agents' usage data for the month.

Agent Health

The Agent Health widget gives you an insight into your agent's health. It displays a time series chart along with counts to show the agents' health data, including `Total Licensed`, `Total Online`, and `Total Offline` for the month.

Deployment Activity

The Deployment Activity widget shows you scheduled deployment counts. It displays a bar chart to show the scheduled deployment jobs count starting with the last 2 days till the next 7 days from the current date.

Current Activity

The Current Activity widget gives you an insight into your active deployments. It displays a table view to give you details on active deployments, that includes deployment information such as `Type`, `Component, Application, or Generic`, `Resource or Environment`, `Process`, and `Start Time`.

Components

Components represent deployable items along with user-defined processes that operate on them, usually by deploying them.

Understanding components is critical to understanding HCL DevOps Deploy (Deploy).

Deployable items or artifacts, can be files, images, databases, configuration materials, or anything else that is associated with a software project. Artifacts can come from a number of sources, such as: file systems, build servers, source version control systems, and repositories. When you create a component, you identify the source and define how the artifacts are imported into Deploy. For example, if the source is Subversion, you specify the Subversion repository that contains the artifacts. Each component represents artifacts from a single source. Sources are defined using source-type plug-ins.

Components have *component processes*. A component process is a series of user-defined steps that operate on the component or its artifacts. Each component has at least one process and can have several. A component process can be a single step or contain numerous relationships, branches, and process switches. Component processes are created with the process editor on page 2398. The process editor is a visual drag-and-drop editor that lets you drag process steps onto the design space and configure them as you go. As more steps are placed, you visually define their relationships with one another. Process steps are selected from a menu of standardized steps. Steps are provided for several utility processes, such as inventory management, and workflow control. More process steps are provided by automation-plug-ins. A component process can have steps from more than one plug-in. See Plug-ins on page 2556.

After you define the source and processes for a component, you import its artifacts into the artifact repository, CodeStation. Artifacts can be imported automatically or manually. By default, a complete copy of the content is imported into CodeStation. The original artifacts are untouched. Each time a component is imported it is assigned a *component version*, which is used to ensure that the appropriate component instances get deployed. Versions can be assigned automatically, applied manually, or from a build server. Every time the artifacts are modified and imported, a new version of the component is created.

You can create *Component templates* that contain processes and configuration properties, which you can use to create components. See Component templates on page 2308.

Creating components

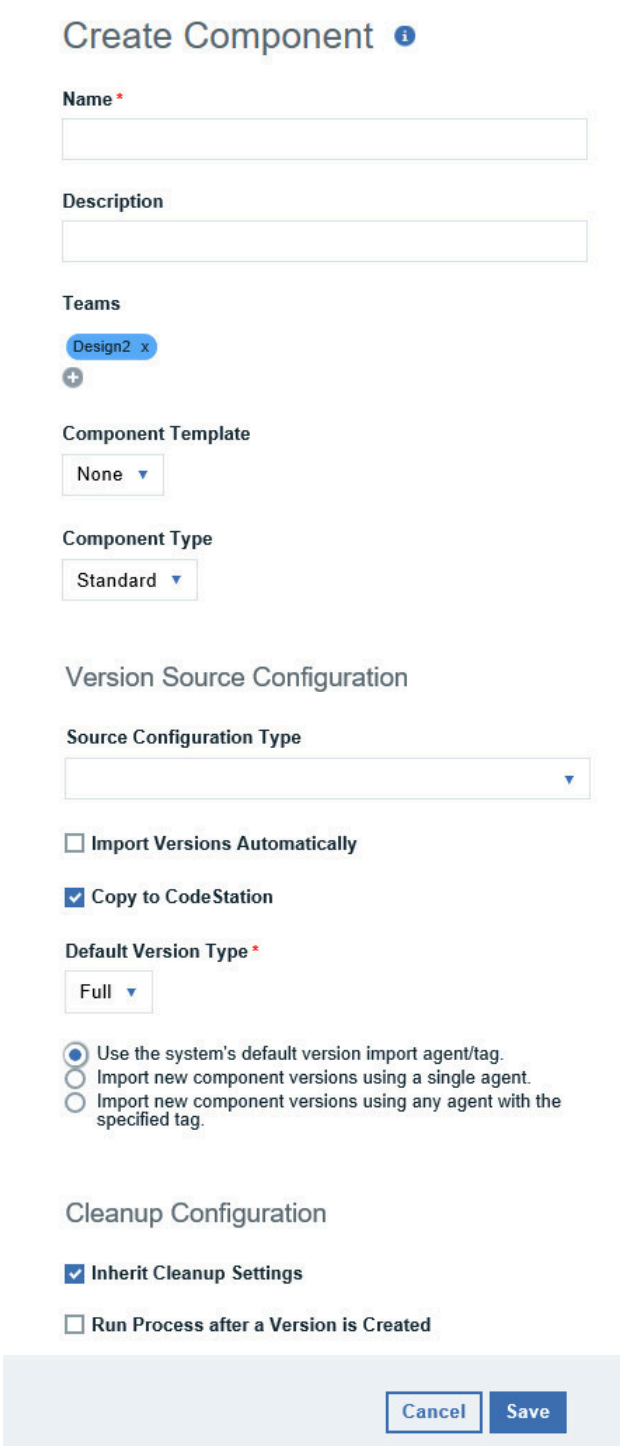
A component consists of artifacts of a single source type and one or more processes. Component creation is the same for all source types, however; the content can be different.

About this task

Components can be created manually, by using a template, or imported. This topic includes the steps for creating a component manually. See Component templates on page 2308 and Importing components on page 2296 for the other methods.

Several fields in the **Create Component** window are the same for every source type, while others depend on the source that is selected from the **Source Configuration Type** list. The graphic below show the **Create Component** window with the common fields for all configuration types.

Some of the configuration options might be restricted based on the role assigned to you. Check with your administrator, if you need a specific option enabled.



1. On the **Components** page, click **Create Component**.
2. To use a template for the new component, select a template from the **Component Template** list.

   When using a template, the component inherits source configuration, properties, and processes from the template. A component can have a single template associated with it. The default value is none.

If you select a template, the **Template Version** field is displayed which is used to select a template version. By controlling the version, you can roll out template changes as needed. The default value is `Latest Version`, which means that the component uses the newest version which is based on creation date.

Note: If you select a template that includes a source type, the dialog box changes to reflect values that are defined for the template. Several fields, including the **Source Configuration Type** field are populated and locked.

For information about templates, see Component templates on page 2308.

3. Type a name and description for the component.
   The name identifies the component throughout the product user interface. The description conveys more information about the component. If the component is used by more than one application, for example, entering `Used in applications A and B` can help identify how the component is used. The **Name** field is required and must contain a value.
4. Specify one or more teams that can manage the component.
   Click **Add Teams** and select from the list of previously defined teams. For information about teams, see Managing Security on page 2149.
5. Select the component type in the **Component Type** field.
   Select **z/OS** if the component is based on IBM® z/OS® artifacts, otherwise use the default value of `Standard`. If you select **z/OS**, the source configuration controls are not available. Artifacts that are stored in z/OS® do not use source configurations. To learn more about configuring and deploying z/OS® components, see Deploying components to the z/OS platform on page 2531.
6. In the **Source Configuration Type** list, select type of source for which the component is being defined. This field defines the source type for the component's artifacts.

   Sources are defined with source configuration plug-ins. If you do not see a desired source type, verify that the source-type plug-in is installed.

   All artifacts must have the same source type. Selecting a source type shows the fields that are associated with the plug-in. Source-dependent fields are used to identify and configure the component's artifacts.
7. Complete common properties that configure how and when artifact versions are imported.

   Import Versions Automatically

   If selected, the source location is periodically polled for new versions. Found versions are automatically imported. The default polling period is 300 seconds, which can be changed with the **Automatic Version Import Check Period** setting on the **System Settings** pane. If left cleared, you can manually create versions by using the **Versions** pane. By default, the box is cleared.

   Copy to CodeStation

   If selected, imported artifacts are copied to a CodeStation repository which is a tamper-proof embedded artifact repository. If cleared, only metadata about the artifacts is imported. For additional information, see System settings on page 2086.

Default Version Type

Defines how component versions are imported into CodeStation. The `Full` value means that the version is comprehensive and contains all artifacts. The `Incremental` value means that the version contains only the artifacts that were modified from the previous version. The default value is `Full`.

8. Select the type of versions to create by default on version imports.
   You can import full version or increment type during version import and specify the additional import configuration:

   Use the system's default version import agent/tag

   If selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. The agent or tagged agents must have access to the system where the artifacts are located. See System settings on page 2086

   Import new component versions using a single agent

   If selected, use the **Agent for Version Imports** field to designate the agent that imports artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from the server location, install an agent in the same location and specify that agent.

   Import new component versions using any agent with the specified tag

   If selected, use the **Agent Tag for Version Imports** field to designate the tag that is used to identify the agents that can import artifacts into the component. Tagged agents must have access to the system where the artifacts are located.

9. Configure common cleanup properties.

   Inherit Cleanup Settings

   Specify whether to use the values specified in the system settings for artifact cleanup or specified them for this component. If this check box is cleared, you must specify values for the following settings:

   Days to Retain Versions

   The number of days to retain component versions after they are created. For example, if you specify 30, component versions are archived when they are older than 30 days and have no other setting that keeps them. To retain component versions indefinitely, enter - 1. To use the system default retention value, enter 0.

   Number of Versions to Retain

   The number of versions to retain for each component. For example, if you specify 10, the server keeps the 10 most recently created component versions. To retain all component versions, enter -1. To use the system default retention value, enter 0.

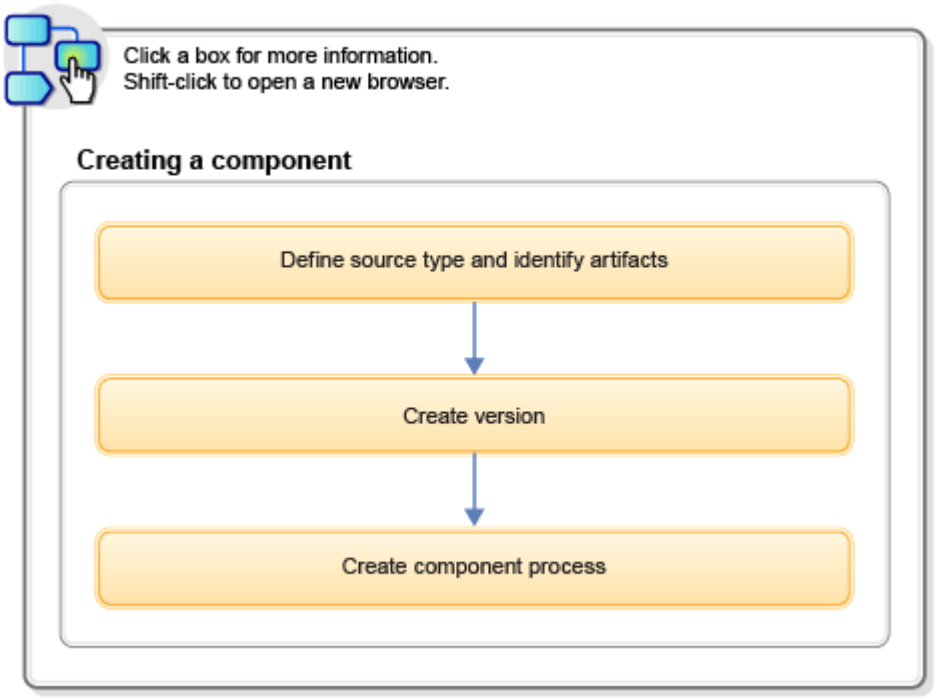
   For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

10. Complete properties based on the source configuration type selected.
    Each source configuration has a unique set of required and optional properties that are displayed after the configuration type is selected. For details on these properties, refer to the plug-in used to integrate the source type.
11. Click **Save**.
    Saved components are listed in the **Components** pane.

Getting started with creating components

Components are the centerpiece of the HCL DevOps Deploy (Deploy) deployment engine. Components associate deployable items (artifacts) with processes that deploy them.

The following diagram summarizes the basic steps for creating components:



1. Click this area for information about defining components. on page 2248
2. Click this area for information about importing component versions. on page 2301
3. Click this area for information about creating component processes. on page 2299

The source type identifies the artifacts that the component uses. The source type can be almost anything that is associated with a software project. Sources are defined with *source plug-ins*.

Component versions are created when artifacts are imported into the artifact repository, CodeStation. Version history is maintained and components can be rolled back to earlier versions.

Component processes consist of user-configured steps that operate on components, typically by deploying them. Process steps are provided by *automation plug-ins*. Processes can be created with the process design editor or by using the command-line interface.

Related information

Manual application tasks on page 2387

Installing plug-ins from the plug-in site on page 2559

Component templates on page 2308

Importing components on page 2296

Creating components from the file system

You can create components by importing artifacts from the file system on the HCL DevOps Deploy (Deploy) server. You can import a single version or automatically import versions as they are added to the file system.

About this task

Deploy imports local files by two methods:

File System

This method imports all files in the specified directory. You can manually update these files to create component versions, but the server does not create versions automatically. See Creating components from a nonversioned file system on page 2253.

File System (Versioned)

This method treats each subdirectory in the target directory as a different version of the artifacts. You can set the server to check for new subdirectories periodically and automatically create component versions when new subdirectories are detected. See Creating components from a versioned file system on page 2256.

In either case, the artifacts must be on the same file system as the Deploy server.

Creating components from a nonversioned file system

To import artifacts that are not stored in versioned folders, store the artifacts in a folder and create a component with the **File System** source configuration type.

About this task

When you use this type of component, you must manually create versions. To create versions of components automatically, see Creating components from a versioned file system on page 2256.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **File System**.
6. Specify the following parameters for the component:

   Base Path

   Specify the path to the directory that contains the artifacts.

Always Use Name Pattern

If you select this check box, new versions are automatically named according to the pattern in the **Version Name Pattern** field. If you clear this check box, you must specify a name each time you create a version.

Version Name Pattern

Specify a pattern for each version name. You can use the *${version}* variable to include an automatically incremented version number. For example, by using the `mycomp_${version}` pattern, the server produces versions such as `mycomp_1` and `mycomp_2`.

This parameter is meaningful only if you select the **Always Use Name Pattern** check box.

Next Version Number

Specify the version number for the next version.

This parameter is meaningful only if you select the **Always Use Name Pattern** check box.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

This parameter has no effect on this source configuration type. You must import new versions manually.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

   Result

   The new component is included in the list of components.
8. Import the first version of the component:
   a. Go to the **Versions** tab.
   b. Click **Import New Versions**.
   c. Specify a name for the new version and then click **Save**.

   Result

   The new version is included in the list of versions. You can confirm that the artifacts are included in the version by clicking the version and looking at the list of artifacts.
9. To import future versions of the component, click **Import New Versions** again and specify a name for the new version.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from a versioned file system

To import artifacts with versions, create a directory with subdirectories that represent the versions of the artifacts. Then, create the component with the **File System (Versioned)** source configuration type.

Before you begin

To create this type of component, first arrange the artifacts inside a parent folder for the component and then arrange the artifacts inside subfolders for each version of the component. For example, the following file structure shows a component with three versions and two files in each version:

```
myComponent/
  1.0/
    fileA.txt
    fileB.txt
  2.0/
    fileA.txt
    fileC.txt
  experimental/
    fileA.txt
    fileD.txt
```

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **File System (Versioned)**.
6. Specify the following parameters for the component:

   Base Path

   Specify the path to the directory that contains the artifacts. The content of each subdirectory within this base directory is considered a distinct component version. The subdirectory with the most recent time stamp is considered to be the latest version.

   Preserve Execute Permissions

   For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

   Text File Extensions

   If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

   ```
   txt,log,ini,sh
   ```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from source-code management systems

You can create components from the artifacts in various source-code management systems.

Creating components from Git

Git is a source-code management system.

Before you begin

Install Git on the HCL DevOps Deploy (Deploy) server. When you create the component, you must specify the path to the Git executable file.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **Git**.
6. Specify the following parameters for the component:

   Repository URL

   The location of the Git repository, such as `https://git.example.com/myproject.git`.

   Branch

   The name of the branch to use.

   User Name

   The user name to authenticate with the Git repository if using HTTPS protocol.

   Password

   The password to authenticate with the Git repository if using HTTPS protocol. You can also use the Git Personal Access Token (PAT) if the PAT is defined in the key-value store of the Vault.

   Watch for tags

   Select this check box to use tags as the basis for new component versions.

Includes

A list of file patterns to include in the component. Separate each pattern with a carriage return.

Excludes

A list of file patterns to exclude from the component. Separate each pattern with a carriage return.

GIT Path

The path to the Git executable file on the Deploy server. If you added the Git executable to the system PATH variable, you can specify the name of the executable, such as `git`. If you did not add the Git executable to the system PATH variable, specify the complete path to the Git executable, such as `/usr/bin/git`.

Preserve Execute Permissions

For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from Subversion

Subversion is a source-code management system.

Before you begin

Install the Subversion client on the HCL DevOps Deploy (Deploy) server. For example, on Linux™, you can use the package manager to install Subversion with a command that is similar to these examples:

```
yum install subversion
```

```
apt-get install subversion
```

```
zypper install subversion
```

All Subversion server configurations are supported.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **Subversion**.
6. Specify the following parameters for the component:

   Repository URL

   The location of the Subversion repository, such as

   ```
   http://svn.example.com/svn/myrepository
   ```

   Watch for tags

   Select this check box to use tags as the basis for new component versions.

   Includes

   A list of file patterns to include in the component. Separate each pattern with a carriage return.

   Excludes

   A list of file patterns to exclude from the component. Separate each pattern with a carriage return.

   User

   The user name for the repository.

   Password

   The password for the repository.

   SVN Path

   The path to the `svn` executable file on the Deploy server. If you added the executable file to the system PATH variable, you can specify the name of the executable file, such as `svn`. If you did not add the executable file to the system PATH variable, specify the complete path, such as `/usr/bin/svn`.

   Preserve Execute Permissions

   For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

   Extensions of files to Convert

   If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

   ```
   txt,log,ini,sh
   ```

Proxy Hostname

If you connect to the Subversion repository through a proxy server, specify the host name or IP address of the proxy server.

Proxy Port Number

Specify the port number that is used by the proxy server for client connections.

Proxy Username

If the proxy server requires authentication, specify the user name.

Proxy Password

If the proxy server requires authentication, specify the password.

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component* > Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from ClearCase

ClearCase is a source-code management system.

Before you begin

Install an agent that has access to the ClearCase version object base (VOB) repository.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **ClearCase**.
6. Specify the following parameters for the component:

   Path to location inside the View

   The location of the base directory on the IBM® Rational® ClearCase® server from which files are imported.

   Includes

   A list of file patterns to include in the component. Separate each pattern with a carriage return.

Excludes

A list of file patterns to exclude from the component. Separate each pattern with a carriage return.

Snapshot View

Check this box if the connection is a snapshot view.

Clear Tool Path

If the `cleartool` file is not on the **path** variable, type the location of the `cleartool` executable file.

Preserve Execute Permissions

For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from Microsoft™ Team Foundation Server SCM

Microsoft™ Team Foundation Server SCM is a source code management tool.

Before you begin

You must have performed the following tasks:

- Installed an agent on a computer that is running Microsoft™ Windows™. The agent computer must have access to the Microsoft™ Team Foundation Server. Ensure that Microsoft™ .NET Framework 3.5 is installed on the agent computer.

  Note: The agent must be installed on a computer that running is Windows™.

- Installed TFS Source Config plug-in from the HCL DevOps Deploy (Deploy)HCL DevOps Deploy plug-in site

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.

In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.

3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **TFS_SCM**.
6. Specify the following parameters for the component:

Repository URL

The URL of the TFS repository, such as `http://tfs2010:8080/tfs/mygroup`.

User

The user name of the TFS-credentialed user.

Password

The password of the TFS-credentialed user.

tf.exe Path

The location of the `tf.exe` file, for example, `C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE`.

Workspace

The name of the TFS workspace. A workspace maps local and version-controlled folders on the TFS server.

Server Folder

The location of the TFS server folder.

Local Folder

The location of the workspace local folder.

Preserve Execute Permissions

For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from build integration tools

You can create components from the artifacts in various build integration tools.

Creating components from DevOps Build

You can create components from IBM DevOps Build (Build), a build integration tool.

Before you begin

Install an agent that has access to the Build server.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **DevOps Build**.
6. Specify the following parameters for the component:

   DevOps Build Url

   The URL of the Build server, such as `https://devopsbuild.example.com`.

   User

   The user name of the Build-credentialed user.

   Password

   The password of the Build-credentialed user.

   Project

   The name of the project. Build projects roughly correspond to Deploy components.

   Build Process

   The name of the build process that contains the builds to import.

   Status

   The status that a build life must have before it can be imported. The server only imports build lives that contain this status.

   Artifact Sets

   If **Copy to CodeStation** is checked, type the list of artifact sets to resolve in CodeStation. Place each list item on a new line.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from Apache Maven

Apache Maven is a build integration tool for Java™ projects.

Before you begin

Install an agent that has access to the Apache Maven repository.

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **Maven**.
6. Specify the following parameters for the component:

   Repository URL

   The URL of the Maven repository, for example:

   ```
   https://MyMavenRepo.MyCompany.om/repository/
   ```

   HTTP Proxy Host

   The proxy host for the Maven repository.

   HTTP Proxy Port

   The proxy port for the Maven repository. You must supply the proxy port if you supply a proxy host.

HTTP Proxy Username

The user name for an account that can access the HTTP proxy.

HTTP Proxy Password

The password for an account that can access the HTTP proxy.

User

The user name for the Maven-credentialed user.

Password

The password for the Maven-credentialed user.

Group ID

The name of the `groupID` as defined in the Maven `pom.xml` file. The group ID identifies the project.

Artifact ID

Name of the `artifactID` as defined in the Maven `pom.xml` file. The artifact ID is the name of the unversioned JAR file.

Maven Qualifer

The optional qualifier is appended to the file name after the version number.

File Extension

The file extension for the file type to import.

Latest Build Count

The number of the builds to import. The default value is 1.

Version Name Pattern

The range of build versions to import. Versions are defined by regular expressions. For example, to import all 1.1.*, and 1.2.* versions, you might type `(1.[0-9]+).*`. A number of versions that is equal to the value in the **Latest Build Count** field is imported for each matching version.

Preserve Execute Permissions

For Linux™ and UNIX™ operating systems, select this check box to retain the execute permissions for each file.

Extensions of files to Convert

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component* > Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating components from Microsoft™ Team Foundation Server

Microsoft™ Team Foundation Server is a build integration tool.

Before you begin

You must have completed the following tasks:

- Installed an agent on a computer that is running Microsoft™ Windows™. The agent computer must have access to the Microsoft™ Team Foundation Server. Microsoft™ .NET Framework 3.5 must be installed on the agent computer.

  Note: The agent must be installed on a computer that running is Windows™.

- Installed Microsoft TFS plug-in from the HCL DevOps Deploy (Deploy)HCL DevOps Deploy plug-in site

1. In Deploy, click **Components**, and then click **Create Component**.
   The **Create Component** window displays.
2. To use a template for the new component, select a template from the **Template** list.
   In this case, the component inherits source configuration, properties, and processes from the template. For information about templates, see Component templates on page 2308.
3. Specify a name and description for the component.
4. In the **Teams** fields, specify the access information for the new component.
5. In the **Source Configuration Type** list, select **TFS**.
6. Specify the following parameters for the component:

   TFS URL

   The URL of the TFS server, such as `http://tfs2010:8080/tfs/mygroup`.

   TFS Version

   Select the installed version of the TFS server.

   TFS Username

   The user name of the TFS-credentialed user.

   TFS Password

   The password of the TFS-credentialed user.

TFS Team Project

The name of the project. Projects are collections of items that track related work.

Build Definition

The name of the build to import.

Build Status

Select the status that a build must have before it can be imported. The server only imports builds with this status.

Build Quality

To limit imports to builds with a specific build quality (or build rating), supply the rating. For example, type `Self Test with Work arounds`.

Directory Offset

The offset is relative to the base directory.

File Includes

A list of file patterns to include in the component. Separate each pattern with a carriage return. To include all files, type `**/*`.

File Excludes

A list of file patterns to exclude from the component. Separate each pattern with a carriage return.

Text File Extensions

If text-type files must be converted into another character set, type the list of file extensions to be converted. Matching file types are converted into the default or system character set of the system where the agent is located. Separate list items with commas, for example:

```
txt,log,ini,sh
```

Import Versions Automatically

Select this check box to check for new versions periodically and to import new versions automatically. If you do not import versions automatically, you can import versions manually by clicking **Components > *selected component*** **> Versions** and then clicking **Import New Versions**.

Copy to CodeStation

This option, which is selected by default, creates tamper-protected copies of the artifacts and stores them in the embedded artifact management system, CodeStation. If the check box is cleared, only metadata about the artifacts is imported. In most cases, keep this check box selected. The default for this setting is in the system setting **Components copy to CodeStation by default**. For more information, see System settings on page 2086.

Default Version Type

Specify how to import versions into CodeStation:

Full

Each version is comprehensive and contains all artifacts.

Incremental

Each version contains a subset of artifacts.

Use the system's default version import agent/tag

If this parameter is selected, the agent or tag that you specified on the **System Settings** page is used to import component artifacts. Agents must have access to the system where the artifacts are located. For more information, see System settings on page 2086.

Import new component versions using a single agent

If this parameter is selected, use the **Agent for Version Imports** field to identify the agent that is used to import artifacts into the component. The agent must have access to the system where the artifacts are located. To import artifacts from where the server is located, install an agent in the same location and specify that agent.

Import new component versions using any agent with the specified tag

If this parameter is selected, use the **Agent Tag for Version Imports** field to identify the tag that is used to select agents to import artifacts into the component. All tagged agents must have access to the system where the artifacts are located.

Inherit Cleanup Settings

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Inherit Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

7. Click **Save**.

Results

The new component is listed on the **Components** page. The **Versions** tab shows the available versions of the component. If you selected the **Import Versions Automatically** check box, versions are displayed automatically. Otherwise, click **Import New Versions** to show the available versions. Depending on the number and size of the artifacts, it might take time before the versions are displayed. To see the artifacts in the component, click a component version.

What to do next

Create processes for the component. See Creating component processes on page 2425.

Creating component versions with the command-line client

If HCL DevOps Deploy (Deploy) does not have a source configuration type for your external tool, you can use the udclient command-line client to manage components.

Before you begin

To install the client, see Installing the command-line client on page 2592.

About this task

When you use the udclient client in this way, you can push new versions of components to the server, start processes, add links to external sources, and run many other tasks on the Deploy server as part of your build or SCM publishing process. In this way, you can work with your components in Deploy without changing your current technology.

Follow these general steps to set up a component that uses the command-line client:

1. Create one or more components in Deploy to represent the components of your build. Use the following parameters in the **Create Component** window:
   - In the **Source Configuration Type** field, select **None**.
   - Clear the **Import Versions Automatically** check box.
   - Complete the rest of the fields in the window as usual.

   You can also use the udclient client to create components, but because component creation is a one-time process, components are typically created withe the Web UI.
2. In the scripts for your component, add code that calls udclient client commands to create a component version and upload the artifacts to the Deploy server.
   In most cases, you use several different commands to create the version. The following steps show examples of some of the commands you might use.

   Note: These examples use code in an Apache Ant build file. Specifically, they call the udclient client with the Ant <exec> command, which calls external programs. If you are using a different build or SCM technology, you can adapt the examples to your specific technology to call the udclient client.

   a. Use the createVersion on page 2680 command to create the new version.
      Result
      The following example code creates a version of a component that is named `mycomponent`. The version name is set to the timestamp of the build. The variables `${udclient.weburl}`, `${udclient.username}`, and `${udclient.password}` represent the URL of the server and the user name and password for the server.

```
<exec executable="/bin/bash"
  outputproperty="mycomponent.component.output">

  <arg value="/opt/udclient/udclient"/>

  <arg value="-weburl"/>
  <arg value="${udclient.weburl}"/>

  <arg value="-username"/>
  <arg value="${udclient.username}"/>

  <arg value="-password"/>
  <arg value="${udclient.password}"/>

  <arg value="createVersion"/>

  <arg value="-component"/>
  <arg value="mycomponent"/>
```

```
  <arg value="-name"/>
  <arg value="${buildTimestamp}"/>

</exec>
```

b. Use the addVersionFiles on page 2612 command to upload the artifacts to the component version.

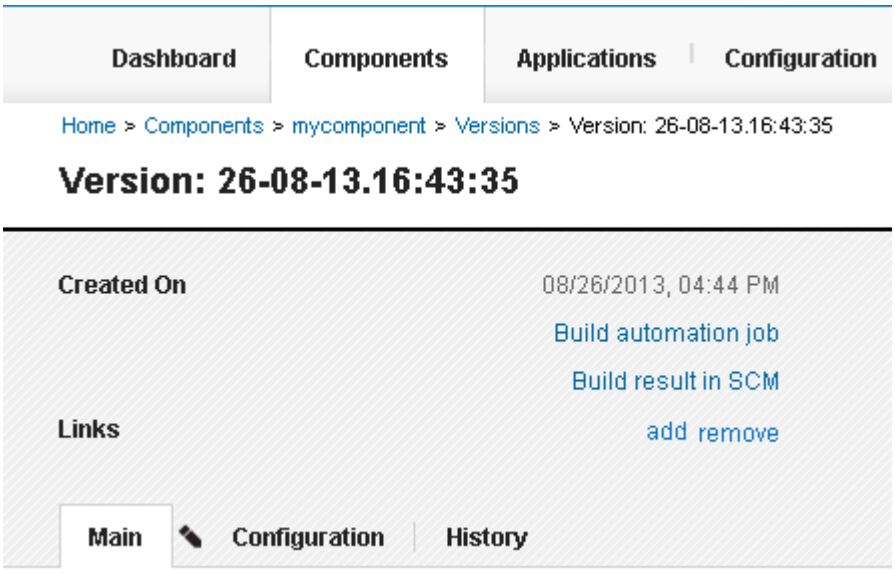
Example

The following example code uploads the files in the `${destdir}` folder to the version:

```
<exec executable="/bin/bash">

  <arg value="/opt/udclient/udclient"/>

  <arg value="-weburl"/>
  <arg value="${udclient.weburl}"/>

  <arg value="-username"/>
  <arg value="${udclient.username}"/>

  <arg value="-password"/>
  <arg value="${udclient.password}"/>

  <arg value="addVersionFiles"/>

  <arg value="-component"/>
  <arg value="mycomponent"/>

  <arg value="-version"/>
  <arg value="${buildTimestamp}"/>

  <arg value="-base"/>
  <arg value="${destdir}"/>

</exec>
```

c. **Optional:** Add links from the component version.

With the addVersionLink on page 2613 command, you can add links from the version to external build resources, such as the page that shows information about the build result. These links appear at the top of the information page for the component version. The following figure shows a component version with a link to the related build job and the related build result in the source-code management system:



Result

For example, the following code adds a link to the build result page at the URL `http://build.example.com/projects/mycomponent/${buildResultUUID}`:

```
<exec executable="/bin/bash">

  <arg value="/opt/udclient/udclient"/>

  <arg value="-weburl"/>
  <arg value="${udclient.weburl}"/>

  <arg value="-username"/>
  <arg value="${udclient.username}"/>

  <arg value="-password"/>
  <arg value="${udclient.password}"/>

  <arg value="addVersionLink"/>

  <arg value="-component"/>
  <arg value="mycomponent"/>

  <arg value="-version"/>
  <arg value="${buildTimestamp}"/>

  <arg value="-linkName"/>
  <arg value="External build result"/>

  <arg value="-link"/>
  <arg value="http://build.example.com/projects/mycomponent/${buildResultUUID}"/>

</exec>
```

Results

Now, when you run a build in your external build tool, the build process creates a component version and uploads the artifacts for the version to the Deploy server. From this point, you can work with the component and version on the server as usual, such as running processes with it.

You can also add other commands to your build script. For example, you can start application processes with the requestApplicationProcess on page 2866 command.

Related information

GitHub Action: Create new DevOps Deploy component version

Creating z/OS® component versions

You can create z/OS component version by running the buztool.sh command from job control language (JCL) or from the z/OS® UNIX™ System Services command line. You can also create z/OS component version by using the z/OS® File Source Config plug-in.

About this task

In an automated system, typically, you write a script that runs the buztool.sh command after a build is created by your build system. The buztool.sh command then creates a component version which can be deployed using a deploy process.

HCL DevOps Deploy (Deploy) server requires the time zone of the z/OS system to display the artifact modified date accurately. Identify the time zone of the z/OS system and set the TZ environment variable. To know more about the TZ environment variable, refer to the z/OS® documentation.

While packaging load modules, System Status Index (SSI) value is converted to epoch time and displayed as Last Modified Date in the component version page. For example, if a load module SSI value is 087ab85c. Converting to decimal format gives the Epoch time as 142260316 which is displayed as July 5, 1974 12:45:16 PM GMT in the component version page. For non-load modules, the buztool.sh command tries to read **ZLM4DATE**, **ZLMTIME** and **ZLMSEC** statistical values from ISPF.

When you create a z/OS component version with an external repository, you will store artifacts in the external repository and store only the metadata of artifacts in Codestation. Hence, when you download the version by using the **Download All** action on the web application, only the metadata XML file, which is stored in Codestation, is displayed in the `packageManifest.xml` file.

What to do next

You can work with the component that contains z/OS® files as you would work with any other component in Deploy. Use the z/OS® Utility plug-in to create processes that deploy components that contain z/OS® files.

See the Deploy website for information about the z/OS® Utility plug-in: Plugins.

Note: IMS.TFORMAT members (member name containing non-readable characters) are not supported. PDS members name can have ONLY [0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@#$] characters.

Creating z/OS® component versions from JCL

To create IBM® z/OS® component versions from job control language (JCL), create and submit JCL to run the buztool.sh command.

About this task

Note: The JCL to run buztool.sh is changed in HCL DevOps Deploy (Deploy) 6.2.5 to use the BPXBATCH utility. The JCL created before version 6.2.5 needs to be updated to follow the format in the new example.

1. Create a JCL file in a partitioned data set and base the file on the sample in the *high_level_qualifier*.SBUZSAMP(BUZRJCL) file.
2. Use the buztool.sh createzosversion command to create the component version or buztool.sh createlinkedversion to create the linked version.

Table 726. Arguments for the buztool command

Parameter	Required	Description
-c, **-component**	true	The component name in Deploy. The component name can contain only letters, numbers, and spaces.
-v, **-version**	false	The name of the version to create. If a version is not specified, a version name is generated from the current time stamp. The version name can contain only letters, numbers, and spaces.
-desc, **-description**	false	Description for the version.
-s, **-shiplist**	true	The location of the ship list file.
-verb, **-verbose**	false	To display a trace log, set this parameter to true. If verbose is specified with no value, it is assumed true. This value should always be set to false in all environments, unless additional logging is needed for a specific need. Setting this to true, can cause out of memory errors, if the log generated is immense.
-o, **-output**	false	The full path name of the file where the output of the version creation is written.
-t, **-type**	false	The type of the version to create. Set to full to create a full version. Default is incremental.
-to, **-token**	false	The token used to authenticate the server. This token can be created by a user who has access to create versions in the corresponding component. If the token

Parameter	Required	Description
		is not provided, buztool takes the token from the `<agent_home>/conf/installed.properties` server.token value to authenticate.
-prop, **-propertiesFilePath**	false	The full path of the buztool properties file. The buztool property file contains `deployType` to `copyType` mappings. Use this parameter to pass details of external repository . See The buztool properties file on page 2286, for more information.
-url, **-serverUrl**	false	The URL of the server in which the version is to be created.
-runtime.deploy	false	When set to false, disables hash calculation of artifacts. Default is true.
-zpv	false	To use v2 package format, set this parameter to v2. If the parameter is not provided, it assumes it to be v1.
-extrepo.url	false	The URL of external repository.
-extrepo.repositoryType	false	The type of external repository. The value can be Nexus, or Artifactory.
-extrepo.repository	false	The name of the repository in Artifactory (or Nexus) to which the version is uploaded.
-extrepo.username	false	The user name of external repository.
-extrepo.password	false	The password of external repository.
-extrepo.apiKey	false	The API key of JFrog Artifactory to be used.
-extrepo.rawDirectory	false	The raw directory in the Nexus repository to be used.
-extrepo.httpProxyHost	false	The proxy host for external repository.
-extrepo.httpProxyPort	false	The proxy port for external repository.
createlinkedversion	true	The argument to create a linked version **Note:** This argument is applicable to z/OS linked versions.
-from.hash	true	Specifies the commit from which a build tool like IBM Dependency Based Build calculates the changed files and performs a build. The system assumes that a previous version already exists for this commit, and Deploy uses that version as the parent for the new version that you create.

Parameter	Required	Description
		Note: This argument is applies only to z/OS® linked versions.
-to.hash	true	Specifies the commit hash in the repository where the build runs, typically the HEAD of the branch. This commit will be mapped to the version that you are creating. **Note:** This argument applies only to z/OS® linked versions.
-parent.version	true	Specifies the parent version name for the version that you are creating. This field is required only when multiple versions exist for the commit ID specified in the `-from.hash` argument. **Note:** This argument applies only to z/OS® linked versions.

Example

For example, the following command creates a new version in the CICSModules component:

```
/opt/install-dir/agent/bin/buztool.sh "createzosversion" "-c" "CICSModules" "-v"
        "fixbug100" "-s" "/u/ucduser/build/shiplist.xml"
```

As shown, you must enclose all arguments in double quotation marks.

3. Modify the sample file to reflect your system configuration.
4. Submit the JCL.

Results

The content that the ship list file specifies is available in a component in Deploy.

Creating z/OS® component versions from z/OS® UNIX™ System Services

To create IBM® z/OS® component versions from z/OS® UNIX™ System Services, submit the buztool.sh command from a z/OS® UNIX™ shell.

1. Open a z/OS® UNIX™ shell.
2. Use the buztool.sh createzosversion command to create the component version or buztool.sh createlinkedversion to create the linked version.

Table 727. Arguments for the buztool command

Parameter	Required	Description
-c, **-component**	true	The component name in Deploy. The component name can contain only letters, numbers, and spaces.
-v, **-version**	false	The name of the version to create. If a version is not specified, a version name is generated from the current time stamp. The version name can contain only letters, numbers, and spaces.
-desc, **-description**	false	Description for the version.
-s, **-shiplist**	true	The location of the ship list file.
-verb, **-verbose**	false	To display a trace log, set this parameter to true. If verbose is specified with no value, it is assumed true. This value should always be set to false in all environments, unless additional logging is needed for a specific need. Setting this to true, can cause out of memory errors, if the log generated is immense.
-o, **-output**	false	The full path name of the file where the output of the version creation is written.
-t, **-type**	false	The type of the version to create. Set to full to create a full version. Default is incremental.
-to, **-token**	false	The token used to authenticate the server. This token can be created by a user who has access to create versions in the corresponding component. If the token is not provided, buztool takes the token from the `<agent_home>/conf/installed.properties>` server.token value to authenticate.
-prop, **-propertiesFilePath**	false	The full path of the buztool properties file. The buztool property file contains `deployType` to `copyType` mappings. Use this parameter to pass details of external repository . See The buztool properties file on page 2286, for more information.
-url, **-serverUrl**	false	The URL of the server in which the version is to be created.
-runtime.deploy	false	When set to false, disables hash calculation of artifacts. Default is true.
-zpv	false	To use v2 package format, set this parameter to v2. If the parameter is not provided, it assumes it to be v1.
-extrepo.url	false	The URL of external repository.
-extrepo.repositoryType	false	The type of external repository. The value can be Nexus, or Artifactory.

Parameter	Required	Description
-extrepo.repository	false	The name of the repository in Artifactory (or Nexus) to which the version is uploaded.
-extrepo.username	false	The user name of external repository.
-extrepo.password	false	The password of external repository.
-extrepo.apiKey	false	The API key of JFrog Artifactory to be used.
-extrepo.rawDirectory	false	The raw directory in the Nexus repository to be used.
-extrepo.httpProxyHost	false	The proxy host for external repository.
-extrepo.httpProxyPort	false	The proxy port for external repository.
createlinkedversion	true	The argument to create a linked version **Note:** This argument is applicable to z/OS linked versions.
-from.hash	true	Specifies the commit from which a build tool like IBM Dependency Based Build calculates the changed files and performs a build. The system assumes that a previous version already exists for this commit, and Deploy uses that version as the parent for the new version that you create. **Note:** This argument is applies only to z/OS® linked versions.
-to.hash	true	Specifies the commit hash in the repository where the build runs, typically the HEAD of the branch. This commit will be mapped to the version that you are creating. **Note:** This argument applies only to z/OS® linked versions.
-parent.version	true	Specifies the parent version name for the version that you are creating. This field is required only when multiple versions exist for the commit ID specified in the `-from.hash` argument.

Parameter	Required	Description
		Note: This argument applies only to z/OS® linked versions.

Example

For example, the following command creates a new version in the CICSModules component:

```
buztool.sh createzosversion -c CICSModules -v fixbug100 -s /u/ucduser/build/shiplist.xml
```

If an argument contains spaces, enclose the argument in double quotation marks, as shown in the following example:

```
buztool.sh createzosversion -c "CICS Modules" -v "fixbug 100" -s /u/ucduser/build/shiplist.xml
```

Results

The content that the ship list file specifies is available in a component in Deploy.

Note: If the agent is upgraded from V7.0.5, the postfix "/artifactory/" is not automatically added to the Artifactory URL from agent V7.1.0. This may result in HTTP response error code 404 while creating version using the buztool. Add postfix "/artifactory" manually to the *url* property in the artifactory properties file to avoid the 404 HTTP error.

Creating z/OS® component version using the v2 package format

You can create IBM® z/OS® component versions from a new package version called version 2 or v2. You can create v2 component versions using the buztool.sh command.

Currently, the artifacts are compressed at the partitioned data set (PDS) level and stored as binary files in Codestation, Nexus, or Artifactory. This condition makes HCL DevOps Deploy (Deploy) z/OS® versions unique because distributed systems packages are stored at the artifact level and not at the folder level.

The `cp` command is enhanced in USS to provide a way to copy artifacts to and from MVS without losing meta relations like ALIAS. The Deploy v2 package format utilizes the enhanced `cp` command to package versions as-is and store them in Codestation, Nexus, or Artifactory.

You can benefit from several advantages by using the v2 package format:

- Component version artifacts are downloaded from Codestation, Nexus, or Artifactory and viewed in an editor that supports viewing files in mainframe encoding. Artifacts are downloaded in native encoding that IBM® z/OS® uses rather than in ASCII.
- During deployments, the token replacements are done in USS because these files are extracted in USS. This way, you deploy the token replaced artifacts directly into the target environment.
- Merging component versions using the v2 package format has improved performances than merging in previous format

Note:

- The current packaging format is still supported. The v2 package format is optional.
- For better results, move components one at a time to the new package format. After Deploy component starts using v2 package format, stay in that format for best results.
- No change is required to the component process design to deploy component versions with the new package format.
- No change is required to the rollback process.
- Starting from version 7.1.0, you can merge component versions within the same package version. Note that merging is not supported between the v1 and v2 formats.
- Datasets with spanned records are not supported.

You can create component versions from the v2 package by using either of these methods:

- In the `buztool` command, use the `createzosversion2` option instead of `createzosversion`.
- Use the `-zpv v2` buztool parameter.

For example, use either of the following commands to create a new component version in the CICSModules component:

```
buztool.sh createzosversion2 -c CICSModules -v fixbug100 -s /u/deployuser/build/shiplist.xml
```

```
buztool.sh createzosversion -zpv v2 -c CICSModules -v fixbug100 -s /u/deployuser/build/shiplist.xml
```

Note:

- Enclose arguments, that have space, in double quotation marks.
- You can use additional parameters, if needed. The parameters are described in Creating z/OS component versions from z/OS UNIX System Services on page 2282

The buztool properties file

Copying files from MVS to USS and back requires that you know the content type of the artifact. Deploy must determine whether the artifact is an executable file (like a loadlib file) or a binary file (like a DBRM file), or a text file (like a COBOL program). To do this, a property file must be passed as input to the `buztool` command with the `-prop <property-file>` parameter. This file contains the `deployType` to `copyType` mapping. Any number of unique deploy types can be mapped to any of the copy types: load, binary, or text. Deploy types must be prefixed with the `deploytype` parameter. Uncomment the parameters you want to use.

By default, container level deployType attributes in shiplist XML are mapped to below copy types :

```
LOAD -> Executable
OBJ (or) DBRM -> Binary
TEXT -> Text
```

Note: The values of **EXTREPO.PASSWORD**, **EXTREPO.APIKEY**, and **SERVER.TOKEN** are encrypted in the buztool.sh script. Change the properties file permission to write during the first run of buztool.sh script. You can set the file permission back to read-only later on for security reasons.

Below is an example of the buztool properties file:

```
# This property file contains samples for following
#
#   1) Defining Deploy Type and Copy Type mappings
#    (which will be later used to create version in new package format)
#
#   2) Setting Runtime Delta Deployment Flag
#
#   3) Defining External Repository details
#
#   4) Declaring alternate Deploy Server details for creating version
#
# * All the lines in the properties file starting with "#" are comments
#   and can be kept as is for reference.
#
#   Please find detailed explaination for each section below
#
#
# Section - 1 :
# ~~~~~~~~~~~~
#
# Defining Deploy Type and Copy Type mappings to create version in
# new package format.
#
# * Deploytypes are mandatory for every Dataset Container in shiplist
#   to create version in new package format.
# * Deploytypes must be prefixed with "deploytype."
#
# Example:
# -------
#
# If deployType is "CICSDBRM", a BINARY dataset then add entry as below
# deploytype.CICSDBRM=BINARY
#
# If deployType is "CICSLOAD", a EXECUTABLE then add entry as below
# deploytype.CICSLOAD=EXEC
#
# If deployType is "CICSLIB", a TEXT dataset then add entry as below
# deploytype.CICSLIB=TEXT
#
# * Property file will have below entries.
#
# deploytype.CICSDBRM=BINARY
# deploytype.CICSLOAD=EXEC
# deploytype.CICSLIB=TEXT
#
#  Add deploytypes to the list as needed.
#
# NOTE:
# ----
# * DeployType is unique in the property file
```

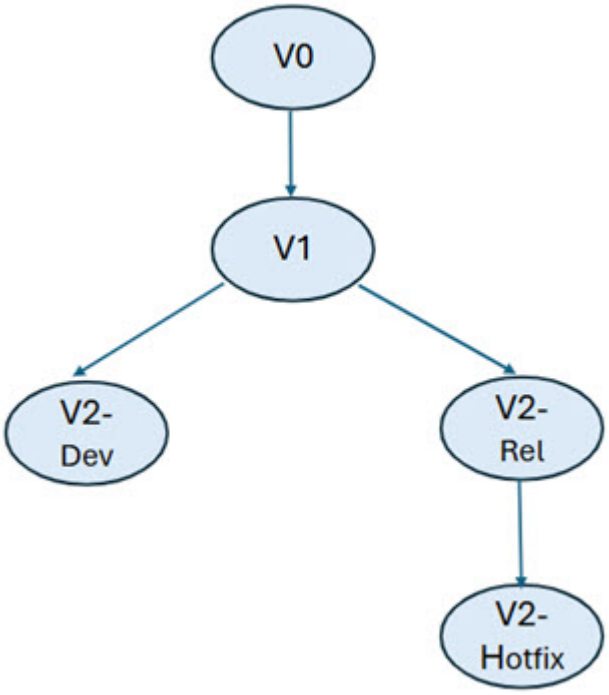
```
#   i.e., each deployType must be assigned to only one copyType.
#
deploytype.JCL=TEXT
# Above entry is used to define deploytype with JCL to copyType as TEXT
# to create a version in new package format from below script
# /var/devops-deploy/agents/deployuser/v7102/installTest/zossamples/test-create-version-new-format.sh
#
#
# Section - 2 :
# ~~~~~~~~~~~~
#
# Setting Runtime Delta Deployment Flag.
#
# Example:
#
# Defining property as below skips the Runtime Delta Deployment
# calculations Logic.
# RUNTIME.DELTADEPLOY=FALSE
#
# NOTE:
# ----
# * Default it is considered as TRUE for version creation.
#
# Section - 3 :
# ~~~~~~~~~~~~
# Defining details to store version artifacts to an external repository.
#
# Example:
#
# # Common external repository inputs
# EXTREPO.REPOSITORY_TYPE={Artifactory/Nexus}
# EXTREPO.USERNAME=
# EXTREPO.PASSWORD=
# EXTREPO.URL=
# EXTREPO.REPOSITORY=
#
# # Additional inputs, if external respository type is "Artifactory"
# EXTREPO.APIKEY=
# EXTREPO.httpProxyPort=
# EXTREPO.httpProxyHost=
#
# # Additional inputs, if external respository type is "Nexus"
# EXTREPO.RAWDIRECTORY=
#
#
# Section - 4 :
# ~~~~~~~~~~~~
#
# Declaring alternate Deploy Server details to create version
#
# Example:
#
# Buztool requires below two properties to create a version on a
# alternate Deploy Server
#
# SERVER.URL=
# SERVER.TOKEN=
#
```

```
# NOTE:
# ----
# * On default, buztool creates version on a Deploy Server defined during
#   installation
#
```

Linked versions in z/OS® components

A *linked version* is a new versioning model for HCL DevOps Deploy (Deploy) z/OS® component types. By using the *linked version*, you can create a sequence of interconnected versions that maintains the logical progression of changes similar to a Git commit history. When you deploy a *linked version* to an environment, it reflects the exact state of the application at that point in the version sequence of the application.

When you create a *linked version*, you must ensure that each version has a preceding version designated as its parent. The first version of the component does not require a parent. This tree-like structure forms a hierarchical sequence of versions, as shown in the following image:



***Linked versions* provides stronger traceability.**

Each *linked version* is created by using a specific GIT commit that you provide as input. This commit reference establishes a continuous chain of custody so that you trace the deployed binary artifacts back to the corresponding source code changes.

This granular traceability is essential for the following purposes:

- **Compliance and auditing**: You can quickly verify the specific code changes that were deployed to an environment.
- **Debugging and rollback**: You can identify the exact changes that introduced a defect, and you can to roll back to the version that corresponds to the preceding stable commit.

By modeling versions in this manner, Deploy ensures that deployed artifacts maintain the same degree of integrity, historical context, and detailed information as the source code. This method provides tight alignment between deployment tracking and established development workflows.

A linked version offers the efficiency of an incremental version while functioning like a *full version*. During deployment, only the delta artifacts are copied instead of copying the entire application, which reduces the processing time and resource usage.

Notes:

- Each component can have only one root-linked version.
- Every linked version that you create after the `root` must be a direct or indirect descendant of that `root` version

Restrictions and limitations: You cannot deploy an *incremental version* on top of a *linked version*; however, you can deploy a *linked version* over an *incremental version*. *Full versions* and *linked versions* are interoperable.

Creating z/OS® component linked versions

You can use the *linked version* type to create a sequence of interconnected versions that maintains the logical progression of changes.

Before you begin

You must have performed the following tasks:

- Installed the z/OS® agent and ensure that it is running.
- Ensured that a valid z/OS® component is available in the server.

Notes:

- The *linked version* type is currently supported for only external GIT repositories. Therefore, you must provide external repository inputs when you create a linked version.
- When you create the first *linked version* for a component, you must not provide the **-from.hash** parameter. The initial *linked version* is designated as the `root` version for the component. It typically serves as the baseline and includes all application binaries after the transition to *linked versions*.

About this task

You can create a linked version by running the buztool.sh command, which is included with the HCL DevOps Deploy (Deploy) z/OS® agent.

Run the buztool.sh command, which is included with the z/OS® Deploy agent package.
Example commands:

- Creating a root version

```
buztool.sh createlinkedversion -v feature-1.0 -c
 MYCOMP -s /u/build/shiplist.xml -prop /u/version.prop -to.hash 2b1820db6bd59dd5301038cb05b8411b81732b17
```

- Creating a child version

```
buztool.sh createlinkedversion -v feature-2.0 -c
 MYCOMP -s /u/build/shiplist.xml -prop /
u/version.prop -from.hash 6f755de14c77c9f688fb51f3801ac1cf4744dd6e
 -to.hash 2b1820db6bd59dd5301038cb05b8411b81732b17
```

Additionally, the buztool.sh arguments required for creating a linked version are as follows:

Buztool argument	Description
createlinkedversion	The argument to create a linked version
-from.hash	Specifies the commit from which a build tool like IBM Dependency Based Build calculates the changed files and performs a build. The system assumes that a previous version already exists for this commit, and Deploy uses that version as the parent for the new version that you create.
-to.hash	Specifies the commit hash in the repository where the build runs, typically the HEAD of the branch. This commit will be mapped to the version that you are creating.
-parent.version	Specifies the parent version name for the version that you are creating. This field is required only when multiple versions exist for the commit ID specified in the `-from.hash` argument.
-v, **-version**	The name of the version to create. If a version is not specified, a version name is generated from the current time stamp. The version name can contain only letters, numbers, and spaces.
-s, **-shiplist**	The location of the ship list file.
-c, **-component**	The component name in Deploy. The component name can contain only letters, numbers, and spaces.

Results

You have created a linked version.

Important:

- You can map a Git repository to one component only. Deploy enforces this rule by preventing you from creating a version in a component when a version for the same commit already exists in another component.
- For linked versions, the `shiplist` that you use to create the version must include a non-empty `deployType` attribute. This attribute classifies the artifacts and enables to process different artifact types separately

during deployment. For example, JCL artifacts might use a `deployType` value of `BATCH`, while `DBRM`artifacts might use `BINDS`.

- The `deployType` value must not contain spaces or special characters. Significantly, the `deployType` determines the directory name where all packaged artifacts for the version are stored. For more information on how this directory structure affects deployment behavior, see z/OS Utility.
- Before you begin using linked versions, you must select a consistent and meaningful naming convention for `deployType` to ensure proper grouping of artifacts.

Notes:

- The resource-level `deployType` takes precedence over the container-level `deployType` in plug-in steps such as **Generate Artifact Information** and **Multiple Generate Artifact Information**.
- You can delete only those versions that do not have child versions from the **Versions** tab of the component.

Example

A sample **shiplist** is as follows:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.JCL" type="PDS" deployType="JCL">
        <resource name="BLZCPBTK" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD1" type="PDS" deployType="CICSLOAD">
        <resource name="*" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD2" type="PDS">
        <resource name="ORDRSET" type="PDSMember" deployType="CICSLOAD"/>
        <resource name="RDBKC01" type="PDSMember" deployType="CICSLOAD"/>
    </container>
</manifest>
```

Related information

Creating z/OS component versions on page 2279

Creating z/OS component versions from z/OS UNIX System Services on page 2282

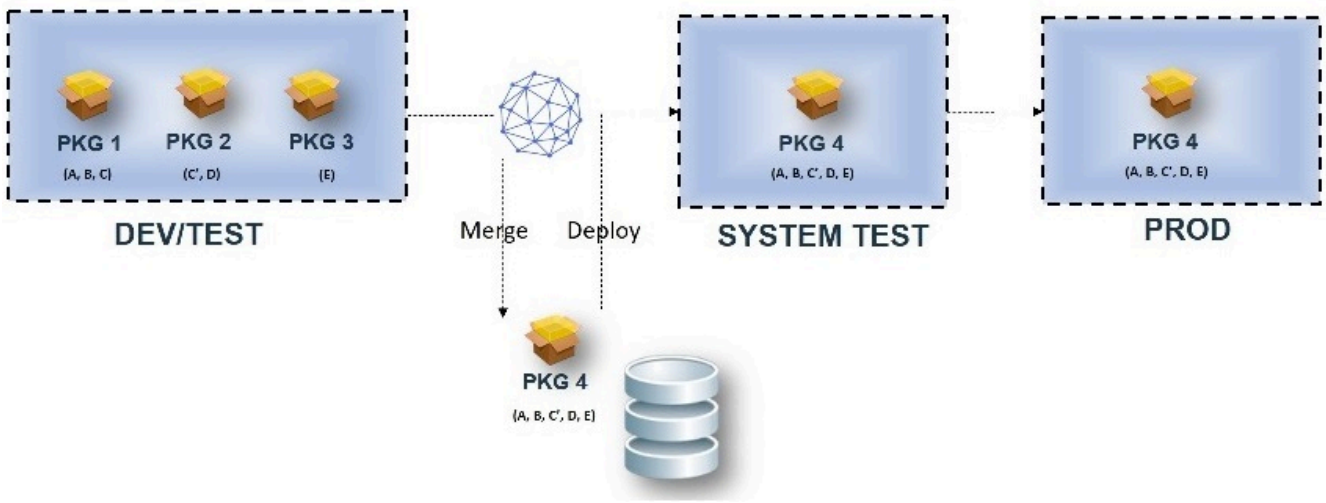
Merging z/OS component versions

Two or more z/OS component versions can be merged from a snapshot into one component version to simplify the deployment to the next environment. Merging component versions in this way avoids deploying intermediate programs into production.

Before you begin

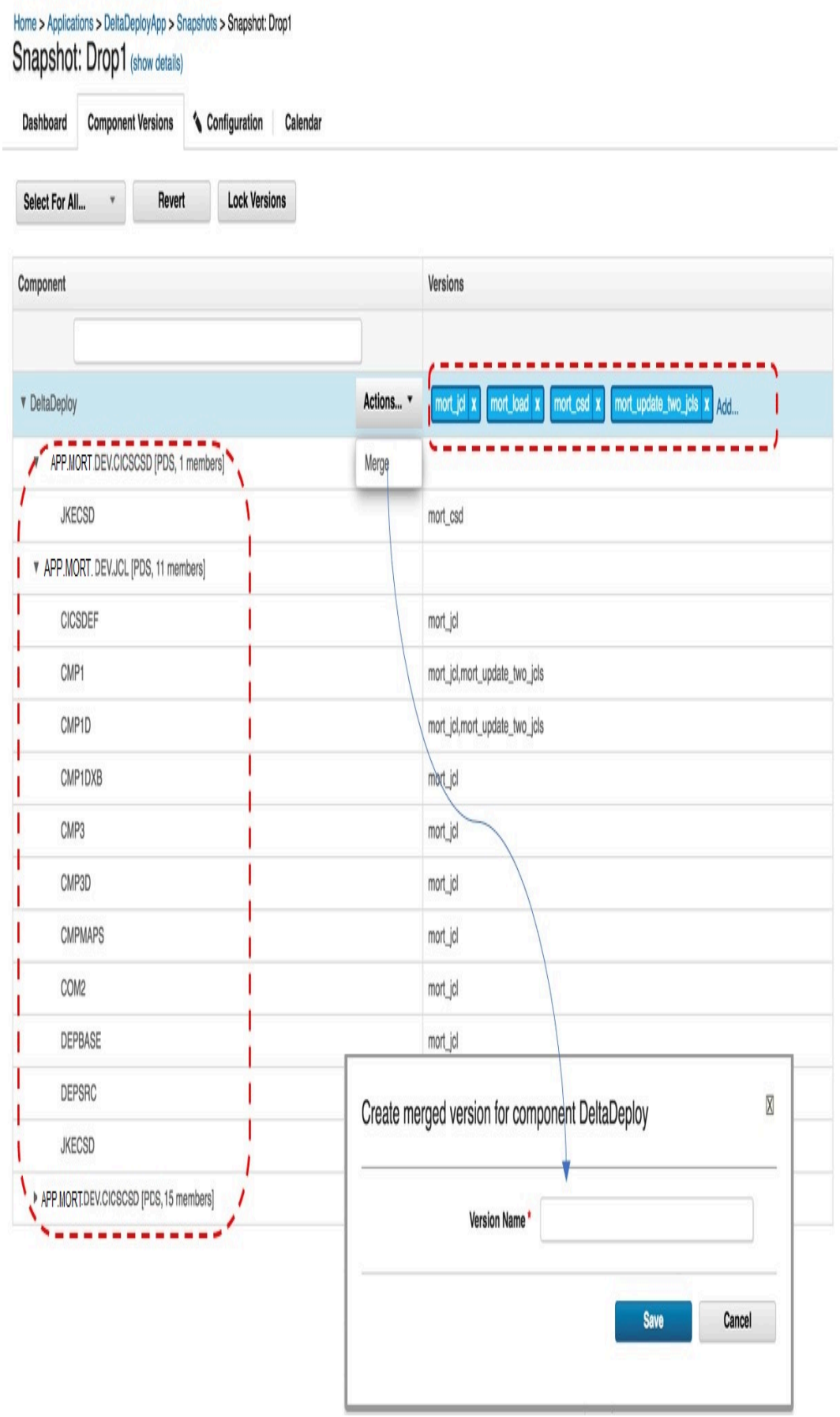
Note the following items when you merge z/OS components.

- Only incremental versions of z/OS components can be merged.
- Only versions stored in the CodeStation can be merged. The merged version is also stored in the CodeStation.
- Merging of versions is not supported when any of the z/OS versions have HFS artifact (shiplist type as directory and file).
- Only data sets are merged into the new version.
- When two or more versions contain the same artifact, the artifact from the latest version in the snapshot is used.
- All customer properties on containers are added to the container in the merged version. This might lead to deployment failures if the deployment process relies on properties on containers.
- The merged version has version properties from the last version in the snapshot.



1. Create a snapshot from an new or existing environment.
2. Add or delete versions in the snapshot's Component Versions view. The **Component** column lists the merged artifacts from all the selected versions. The **Versions** column displays all the versions that contain the artifacts.
3. From the **Actions** menu, click **Merge** for the selected components and supply a name for the merged version. Click **Save**. A new version is created from the merged content of the selected versions.

4. In the **Artifacts** view, click **Show SYS properties**. The source version name and ID are displayed.


Home > Applications > DeltaDeployApp > Snapshots > Snapshot: Drop1
Snapshot: Drop1 (show details)
Dashboard
Component Versions
Configuration
Calendar
Select For All...
Revert
Lock Versions
Component
Versions
DeltaDeploy
Actions...
mort_jcl x
mort_load x
mort_csd x
mort_update_two_jcls x
Add...
Merge
APP.MORT.DEV.CICSCSD [PDS, 1 members]
JKECSD
mort_csd
APP.MORT.DEV.JCL [PDS, 11 members]
CICSDEF
mort_jcl
CMP1
mort_jcl,mort_update_two_jcls
CMP1D
mort_jcl,mort_update_two_jcls
CMP1DXB
mort_jcl
CMP3
mort_jcl
CMP3D
mort_jcl
CMPMAPS
mort_jcl
COM2
mort_jcl
DEPBASE
mort_jcl
DEPSRC
JKECSD
APP.MORT.DEV.CICSCSD [PDS, 15 members]
Create merged version for component DeltaDeploy
Version Name *
Save
Cancel

Results

The following outcomes apply to merged versions.

- Deployment, backup, and restore actions of merged versions proceed like any other version. No change is required in the deployment process.
- Deleting or archiving the original version has no impact on the merged version.
- A merged version can be merged with other versions again.
- Properties that you set on a Partitioned Data Set (PDS) are added from all versions to the PDS in the merged version for the same properties. The later value for the property overwrites the former value.

Note: Overwriting PDS properties might lead to deployment failures if your deployment process relies on the properties that you set in the PDS.

Importing components

Importing and exporting can be especially useful if you have multiple HCL DevOps Deploy (Deploy) servers, for example, and must quickly move or update components.

About this task

When you import components, you can create components or upgrade existing components. Additionally, if the imported components were created from a template, you can use the template or create one.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

Note: If the imported components have the **Import Versions Automatically** parameter set to `true`, the new components automatically import component versions if the artifacts are accessible to the importing server.

To import one or more components:

1. Open the **Import Component** window by clicking **Components > Import Components**.
2. Click **Choose File**, and then select the JSON file that contains the component definition.

   Note: You must create the JSON file that contains the component definition on a server that uses the same version of Deploy as the destination server.

3. To upgrade existing components, select the **Upgrade Component** check box. To create components, leave the check box cleared.
   If the component name in the JSON file (not the name of the file itself) matches an existing component name, then the component parameters are updated. If items such as processes or properties exist, new versions are created. New items, such as processes or properties, are also added. If the name of the component is not found, the import action has no effect.

Note: The component name is the first parameter in the defined file. This example shows the first parameter:

```
"name": "helloWorld",
```

4. If the imported components were created from a template, use the **Component Template Upgrade Type** list to specify how to use the template.
   For these options, the template must be on the importing server. If the imported components were not created from a template, these options are ignored.
   - To use the imported components' template, select **Use Existing Template**. The new components are an exact copy of the imported components. They contain a pointer to the imported components' template. This option is especially useful if you are importing many components that are based on the same template.

     If you are upgrading, the components also point to the imported template.
   - To create a template, select **Create Template**. The new components are an exact copy of the imported components. They contain a pointer to the newly created template, which is based on the imported components' template.

     If you are upgrading components, a template is also created.
   - To ensure that the server uses the newly imported template, select **Fail if Template Exists**. In this case, if the server already contains the template, the import action fails.
   - To create a fresh installation and ensure that a template is not on the importing server, select **Fail if Template Exists**. If you are creating components, the server creates both new components and a template unless the template exists, in which case the components are not imported.

     If you are upgrading components, the upgrade fails if the imported components' template exists.
   - To ensure that the template is on the importing server, select **Fail if Template Does Not Exist**. If you are creating components, it creates both components and a template unless the template does not exist, in which case the components are not imported.

     If you are upgrading components, the upgrade fails if the imported components' template does not exist on the importing server.
   - To upgrade the template, select **Upgrade if Exists**. This option creates components and upgrades the template on the importing server. If the template does not exist, a template is created.
5. If the processes of the imported components call generic processes, use the **Generic Process Upgrade Type** list to specify how the generic processes are upgraded or created. The import process has different results if the generic processes exist or do not exist.

Note: The server tracks generic processes by name. If a generic process on the server is assigned the same name as a generic process that is imported, then the generic process already exists on the server.

- To import a component without updating the generic processes that are on your server, select **Use Existing Generic Process**. If the generic process does not exist, a generic process is created. If a generic process with that name exists, another version is not created.
- To create new generic processes, select **Create Generic Process**. A copy of the generic process is always created. If a generic process with that name exists, a unique string is added to the new generic process's name. When run, the component process calls the new generic process.
- To create a fresh installation and ensure that generic processes are not on the importing server, select **Fail if Generic Process Exists**. If you are creating components, the server creates both new components and new generic processes unless generic processes exist, in which case the import action fails.

  If you are upgrading components, the upgrade fails if the imported components' generic processes exist.
- To ensure that generic processes are on the importing server, select **Fail if Generic Process Does Not Exist**. If you are creating components, it creates both new components and generic processes unless the generic processes do not exist, in which case the import action fails.
- To upgrade the generic processes, select **Upgrade if Generic Process Exists**. This option creates components and new versions of the generic processes on the importing server. If the generic processes do not exist, processes are created.

6. Click **Submit**.

Exporting components

Importing and exporting can be especially useful if you have multiple HCL DevOps Deploy (Deploy) servers, for example, and must quickly move or update components.

About this task

Exporting components creates a JSON file (file extension `json`) that contains the source configuration information, properties, and processes for the components. For more information about JSON, see http://www.json.org/.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

Note: Exporting components in this way includes only the most recent version of each component process.

To export components:

1. On the **Components** pane (**Components**), select the check box next to one or more components.
2. At the top of the table, click **Actions** and then click **Export**.

What to do next

You can load the file into a text editor or save it. Then, you can import the components to another server that uses the same version of Deploy. See Importing applications on page 2347. See Importing components on page 2296.

Component processes

A component process is a series of user-defined steps that operate on a component's artifacts.

Tip: For full coverage of component processes, see Component processes on page 2421 and its subtopics.

Each component has at least one process and can have several. Component processes are created with the process editor. The process editor is a visual drag-and-drop editor that enables you to drag process steps onto the design space and configure them as you go. Process steps are selected from a menu of standard steps. See Processes on page 2398.

HCL DevOps Deploy (Deploy) provides steps for several utility processes such as inventory management and workflow control. More process steps are provided by plug-ins. Deploy provides plug-ins for many common processes, such as downloading and uploading artifacts, and retrieving environment information.

The following process steps are provided by the server. For information about steps that are provided by plug-ins, see the documentation for the individual plug-in in the HCL plug-in site.

A frequently used process can be saved as a template and applied to other components. See Component templates on page 2308.

Importing specific component processes

Component processes define the steps required to deploy, configure, or update a specific component. You can import specific component processes to reuse a standardized deployment across multiple applications or environments without recreating a process each time.

Before you begin

You must have ensured that the component process source file to be imported is a JSON or YAML file.

About this task

By importing processes in HCL DevOps Deploy (Deploy), you can incorporate specific deployment logic for components, making them reusable across different applications and environments.

Note: When you export components, applications, processes, or templates, and then import them on a different computer, you must ensure that you install the same versions of the same plug-ins on both computers.

1. Log in to Deploy.
2. Click **Components**.
   Components page is displayed.
3. Select the component to which you want to import specific component processes.
4. Click **Processes > Import component process**.

The **Import Component process** dialog is displayed.

5. Select **Process Upgrade Type** from the dropdown list.
6. Click **Choose File** to upload the component process source file that you want to import.

Notes:

- You must create the JSON or YAML file that contains the component definition on a server that uses the same version of Deploy as the destination server.
- Only JSON and YAML file types are supported.

7. Click **Submit**.

Results

You have imported the specific component process. The imported component process is listed in **Processes** tab for the component.

What to do next

You can set environment-specific properties by editing the process in the process editor. See Editing processes on page 2453.

Related information

Components on page 2247

Processes on page 2398

Exporting specific component processes

You can export specific component processes to save and share deployment configurations for reuse across applications or environments.

Before you begin

You must have identified the component processes that are to be exported.

About this task

Exporting component processes creates JSON or YAML file (file extension), that contains processes for the components. For more information about JSON, see http://www.json.org/. You can export one or more specific component processes as YAML or JSON files.

Note: When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

1. Log in to Deploy.
2. Click **Components**.
   The **Components** page is displayed.

3. Click the component from which you want to export the specific process.
4. Click **Processes** tab.
   The list of associated process with the selected component is displayed.
5. Export the processes using any of the following methods:.
   - To export processes in bulk:
     a. Select the processes to be exported.
     b. Click **Bulk Actions**.
     c. Select **Export to JSON** or **Export to YAML**.
   - To export a latest promoted process:
     a. Click the menu option of the process to be exported.
     b. Select **Export to JSON** or **Export to YAML**.
   - To export a draft process:
     a. Open a process in the process designer.
     b. Select a draft version, if required.
     c. Click **Export > Export to JSON** or **Export > Export to YAML**.

Results

You have exported the specific component processes to your local system in the selected file format.

What to do next

After exporting the specified component processes, you can import them into other instances for future use.

Related information

Importing specific component processes on page 2299

Processes on page 2398

Component versions

Each time a component's artifacts are imported into the repository, including the first time, a separate version is stored in CodeStation. Versions can be assigned automatically by HCL DevOps Deploy (Deploy), applied manually, or come from a build server.

Every time a component's artifacts are modified and imported, a new version of the component is created and an entry is made in the Component Version Import log. So a component might have several versions in CodeStation, and each version is unique.

Note: By default, the component version import log is cleaned up every 24 hours. Only the most recent 100 entries are maintained in the log after cleanup.

Artifacts can be imported automatically or manually. By default, a complete copy of an artifact's content is imported into CodeStation. When you copy the artifact to CodeStation, the original artifact content is never modified.

A *version* can be full or incremental. A *full version* contains all component artifacts; an *incremental version* contains only artifacts that are different from the previous version. When performing an incremental import you only need to upload only the changed artifacts. Artifacts that are the same as the base version do not need to be uploaded.

Storing artifacts in CodeStation provides several benefits, such as tamper-proof storage, and the ability to review and validate artifacts with the Deploy user interface. But if you have storage concerns or use a tool like Maven, you can limit CodeStation to using references to the artifacts instead of copying them into CodeStation.

Note: Do not give component versions the name `latest`, `latestVersion`, or `newest`. These names are reserved to represent the most recently created version for certain CLI commands, including requestApplicationProcess on page and createSnapshot on page . You cannot use these keywords to represent component versions in the web interface.

Tracking environment inventory for components

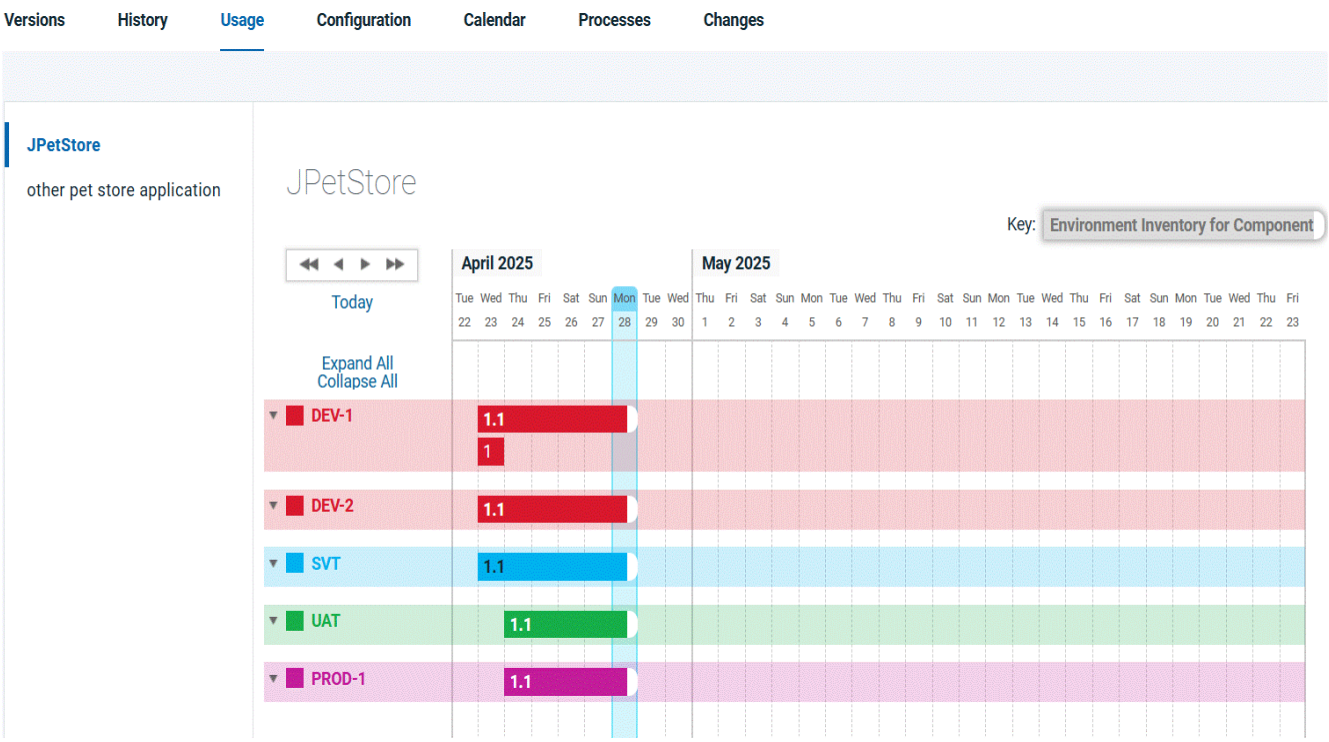
You can track the environment inventory for each component by clicking the component and then going to the **Usage** tab.

About this task

The calendar on this tab shows the deployment request history and state of the selected component. Select an application at the left side of the page, and the calendar shows the deployment history of that component. For each environment, the calendar shows bars that represent the different versions of the component. These bars show when the different versions of the component were installed to that environment or when there was an attempt to install the component to that environment.

Important: The **Usage** tab might not show the version of the component that is deployed. For example, if a deployment fails, the tab includes the component version that was part of the attempted deployment. It does not show the deployed version. Because a component version might not be deployed in an environment, this tab does not show compliancy.

For example, the following figure shows the deployment history for a component, including deployment requests that failed. Two different versions of the component were part of deployment requests to five different environments. The `DEV-1` environment requested the deployment of version 1 of the component on April 23. Then, it requested the deployment of version 1.1 of the component. Because these are incremental versions, the bars overlap when both versions are deployed at the same time.



Importing versions manually

If the component does not import new versions automatically, you can import versions manually.

About this task

To set up a component to import versions manually, see Creating components on page 2248.

Note: Do not give component versions the name `latest`, `latestVersion`, or `newest`. These names are reserved to represent the most recently created version for certain CLI commands, including requestApplicationProcess on page and createSnapshot on page. You cannot use these keywords to represent component versions in the web interface.

1. Display the **Version** pane for the component you want to use (**Components > *selected component* > Versions**).
2. Click **Import New Versions**.
   Depending on the source configuration type, you might see a window that prompts you for more information, such as the name of the version to import.

Results

The new version or versions appear in the table of versions.

Note: If the version import fails, check the logs in version import history.

What to do next

The imported component version can now be compared with an existing version (if any), copied, or deleted. You can also edit the component versions.

Deleting versions

If specific component versions meet the criteria for deletion, you can delete them.

Before you begin

If you want to archive deleted versions, click **Settings > System Settings** and in the **Archive Path** field, specify a folder location for archived versions. Then, click **Save**.

About this task

To delete a version, use the **Delete** action for the version: click **Components > *selected component* > Versions > *selected version* > Delete**. Deleting a version removes the artifacts from CodeStation and the component version from the list of versions.

Note: You cannot delete a component version that was deployed by an application or is used in an application, application snapshot, or resource. If configured, a archive zip will be created and the version will be marked as deleted.

To archive old component versions so that you can restore them later, see Cleaning up component versions on page 2136.

Results

If you specified a folder for archived versions, the deleted version artifacts are in that folder.

Downloading component version artifacts

Artifacts are generated when an application process of a component version runs successfully.

About this task

You can download all artifacts for a component version, or download a delta manifest for linked versions.

1. Display the component version artifacts you want to download (**Components > *selected component* > Versions > *selected version***).
2. Click **Download** and then choose one of the following options to download the artifacts:
   Choose from:
   - Click **Download All** to package all artifacts in one file.
   - For z/OS linked versions,
     - Click **Download Cumulative Manifest** to download an XML that contains the manifest of all artifacts in the application as they existed when the version was created.
     - Click **Download Delta Manifest**, and then select **From Version** from the drop-down to download the delta manifest that shows the differences in artifacts between the current version and the selected version.

Results

A copy of the artifacts is downloaded to your local system.

Adding links to component version

Component version links help you get information about what a particular version contains from various systems of record such as Anthill Pro, Jira, GitHub, and so on.

About this task

You can add links to a component version.

1. Display the component version you want to add links (**Components > *selected component* > Versions > *selected version***).
2. Click **Add**.
   The **Add Link window** is displayed.
3. Enter link name and the URL.
   Optionally, select **Make Priority** to mark the link as favorite.
4. Save your changes.

Results

The link is added to the component version. In case of multiple links, **View All** option is displayed. Using the **View All** option, you can manage the links in the Version Links page.

What to do next

You can make links as favorite by clicking on the star icon next to a link. Favorite links appear first in the list making it easier for you to navigate to the sites.

Component properties

Components can have several types of properties. The server keeps track of changes to component properties as property versions.

To define component properties, click the **Components** tab, click the component, and then go to the **Configuration** tab. The following table describes the types of properties that are available on components.

Table 728. Component properties

Type	Description
Component properties	Component properties are values that are assigned on individual components. Referenced: `${p:component/propertyName}`.
Component properties that	Components inherit custom properties from component templates. These properties cannot be changed on the component. Referenced: `${p:component/custom/propertyName}`.

Table 728. Component properties (continued)

Type	Description
are inherited from component templates	
Component environment properties	Component environment properties are defined on components. However, the property value is not set until the component is deployed to an environment. In this way, a component environment property can have a different value for each deployed instance of the component. However, a value that is set on a component environment property does not override an environment property that has the same name. The environment property has precedence over the component environment property. Referenced: `${p:environment/propertyName}`.
Component version properties	Component version properties are defined on components. However, the property value is not set until a component version is created. In this way, a component version property can have a different value for each component version. Referenced: `${p:version/propertyName}`.
Component resource properties	Component resource properties are defined on components. However, the property value is not set until you map a component to an environment and create a component resource. The property value is set on this component resource. In this way, if you need to deploy multiple copies of a component to the same environment, you can set a different property value for each copy. Note that the component resource property and the resource property definitions are not resolved during deployment when the component is mapped in the Resource Tree using a Component Tag Resource. Component resource properties also generate resource roles on the component resources. Referenced: `${p:resource/propertyName}`.
Component property definitions	Aside from component template properties that become component properties, component templates can also have component property definitions. These properties appear on the component **Basic Settings** tab. Referenced: `${p:component/template/propertyName}`. See Component template properties on page 2310 for more information.

Creating component manual tasks

A component manual task interrupts a component process until manual intervention is done. To create a component manual task, add a manual step to a component process, and select a notification template.

About this task

A process that is interrupted by a manual task remains suspended until the targeted user or users respond. During new process testing, use manual tasks to test the results of steps. After testing, use manual tasks for tasks that are too complicated to automate. Remove manual tasks after testing or when processes are automated.

Similar to approvals, triggered tasks notify targeted users. Affected users can accept or reject the task by using the **Work Items** pane (see Responding to approval requests on page 2394). Unlike approvals, component manual tasks are incorporated within a component process. Do not use a component manual task in place of an approval process. To obtain approval for an entire component, create an approval process instead of a component manual task (see Creating an approval process on page 2388).

1. Open a component process in the process editor.
2. From the **Utility Steps** folder, add a **Manual Task** step to the process.
3. In the **Edit Properties** window, specify the following properties for the step:

Table 729. Component manual task properties

Field	Description
Name	The name for the manual task.
Notification Template	Specify a notification template. The individual tasks map to the notification scheme used by the application (see Creating Notifications in a Notification Scheme on page 2081). If you do not want notifications created for the task, select **None**.
Approver requirements	**Any user** Any user can approve the task. **Only the deploying user** Restricts approval to the user who started the deployment. **Role members by environment** Specify an environment role that can approve the task. **Role members by application** Specify an application role that can approve the task. **Role members by component** Specify a component role that can approve the task.
Add Property	Add one or more properties to the task. You can use these properties later in the process just like output properties from other steps.

4. Click **Save** to save the properties for the step.
5. Add another step such as Shell and add the output properties of manual task to the step to print the approver and approval time information in the output log.

See Manual task output properties on page 2518.

6. Save the process.

Results

When you run the process, it pauses at this step, sends a notification, and waits for approval before the process continues. To approve the task and resume the process, click the **Respond** link for the step in the process log, click **Approve**, and then click **Submit**. You can also resume the process from the **Work Items** tab.

Component templates

With a *component template*, you can save and reuse component processes and properties and create components from them; template-based components inherit the template's properties and process.

Creating component templates

To create a component template, specify information about the template, such as which plug-in it is related to.

To create a template:

1. Display the **Create Component Template** dialog box (**Components > Templates > Create Template**).
2. Enter the template's name in the **Name** field.
3. Enter a description in the **Description** field.
   The description can be used to convey more information about the template.
4. In the **Teams** fields, specify the access information for the template.
5. In the **Component Type** list, select **z/OS** for System z® components and **Standard** for all other components.
6. In the **Source Configuration Type** list, select the source type of the component.
   Selecting a value other than the default `None`, displays more fields that are associated with your selection. Source-dependent fields are used to identify and configure the artifacts. If you select a source, components that are based on the template use the same source. You can also import versions automatically and import configurations. Under the **Import Configuration** field you can create a component template with any of the three options as well as not supplying an option.

   For information about creating components of specific types, see Creating components from the file system on page 2253, Creating components from build integration tools on page 2268, or Creating components from source-code management systems on page 2258.

   Note: If you select a source, any properties you configure are set for any components that are created with the template.

7. Click **Save** to save the template.
   Result
   Saved templates are listed in the **Component Templates** pane.
8. Add components, processes, and properties to the template.

Importing component templates

When you import a template, you can create an entirely new template or upgrade an existing one.

About this task

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

1. Display the **Import Template** dialog (**Components > Templates > Import Template**).
2. Click **Choose File**, and then select the JSON file that contains the template.

   Note: You must create the JSON file that contains the template on a server that uses the same version of HCL DevOps Deploy (Deploy) as the destination server.

3. If you want to upgrade an existing template, check the **Upgrade Template** check box. To create a template, leave the box cleared.
   If the template's name in the JSON file (not the name of the file itself) matches an existing template, the template is upgraded. If the name is not found, the command has no effect.

   Note: The template's name is the first parameter in the defined file. For example,

   ```
   "name": "helloWorldTemplate",
   ```

4. In the **Generic Process Update Type** list, specify how to handle generic processes that are being imported along with the component template and that have the same names as existing generic processes.

   Note: This field affects only generic processes, not component template processes.

   Use Existing Process

   Select this option to ignore generic processes in the imported component template when those processes have the same name as existing generic processes.

   Create Process

   Select this option to rename any generic processes that have the same name as existing processes.

   Fail If Process Exists

   Select this option to cancel the import if the imported component template references any generic processes that have the same name as existing generic processes.

   Fail If Process Does Not Exist

   Select this option to cancel the import if the imported component template references any generic processes that do not have the same name as existing generic processes.

Upgrade If Exists

Select this option to update generic processes to match the imported generic processes.

5. Click **Submit**.

Exporting component templates

Exporting a template creates a JSON file (file extension `json`) that contains the template's configuration information, properties, and processes.

About this task

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

To export a template, on the **Component Templates** pane (**Components > Templates**), click the **Export** link in the **Actions** field. You can load the file into a text editor, or save it. If you save it, a file is created with the same name as the selected component, for example, `helloWorldTemplate.json`.

Component template properties

Component template properties ensure that every component created from a template has the same properties. The types of available properties are described in the following table.

Type	Description
Properties	Custom property. Every component inherits the value that is defined in the template (it cannot be overridden by a component). If you change the value, the change is reflected in components that are created from the template, including components that were previously created.
Component Property Definitions	Every component has this property. It appears on the Create Component dialog box for every component that is created from this template (see Creating components on page 2248). A value that is defined here can be changed by created components.
Environment Property Definitions	Every environment that uses a component that is created by this template has this property. The property appears on the environment's **Component Mappings** pane (**Applications > *selected application* > Environments > *selected environment* > Component Mappings**), see Application environments on page 2354. A value that is defined here can be changed by environment.
Resource Property Definitions	This property appears on resources that the component is deployed to.

Related reference

Component properties on page 2305

Using component templates

When you create a component from a template, the component inherits the template's process and properties, if any.

1. Display the **Create Component** window (**Components > Templates > *selected template* > Create Component**).
   The **Create Component** dialog box (the same dialog box that is used to create non-template-based components) is used to configure components. Properties that are defined in the template are predefined. If a source was selected in the template, the source is set here and the **Source Configuration Type** field is locked. For information about using this dialog, see Creating components on page 2248
2. After you configure editable properties, save the component.

Results

Templates that are used to create components are listed in the **Templates** view.

Components that are created from templates are listed in the **Components** view.

Note: To delete a component template you will need to also delete the components that are using it. The template will be marked as deleted, after deletion, as well as all of it's processes.

Component change logs

Change logs provide information about modifications to components.

To see change details, display the log for a selected component-related activity (**Components > *selected component* > Changes**). Information for any change that triggered a Commit ID is displayed.

Deleting components

When a component is deleted, it cannot be activated later.

To delete a component, use the actions on the **Components** pane for the intended component.

When a component is deleted, it, along with all files for versions of that component, is removed. The component cannot be activated later. The original artifacts are not affected; only CodeStation copies are deleted. Components cannot be deleted if they are used by an application. To delete a component that is used by an application, first remove the component from the application. All deployment triggers from application that references the component will be removed. All inventory for the component will also be removed. All processes of component will be marked as deleted.

Note: Component will not be removed from the database; processes that reference deleted component can still be viewed.

To delete a specific component version, see Deleting versions on page 2304.

Resources

A *resource* is a logical deployment target that typically resolves to an agent and a user-defined construct that is based on the architectural model of HCL DevOps Deploy (Deploy).

Resources aid bookkeeping; inventory is tracked for resources. Resources are created and managed through the user interface.

A resource can represent an agent, agent pool, component, or an organizational entity that is used to group other resources. A component-type resource can point directly to the agent or agent pool that deploys it, or point to another resource in a hierarchical chain of resources. If a resource is part of a hierarchy, it delegates its automation, if any, up the chain until it finds an agent.

A resource represents a deployment target, such as a physical server, virtual machine, database, or Java™ Platform, Enterprise Edition container. Components are deployed to resources by agents (which are physical processes). Resources generally are hosted on the same host where its managing agent runs. A host can have more than one resource. If an agent is configured to handle multiple resources, a separate agent process runs for each one.

A resource can represent a physical server, which is the simplest configuration, or a specific target on a system, such as a database or server. So a host can have several resources represented on it. In addition, a resource can represent a process that is distributed over several physical or virtual machines. Finally, environments consist of resources.

To run a deployment, at least one resource must be defined and (usually) at least one agent. ("Usually" because trivial deployments can be done without an agent.) Typically, each host in a participating environment has an agent running on it to handle the resources that are located there.

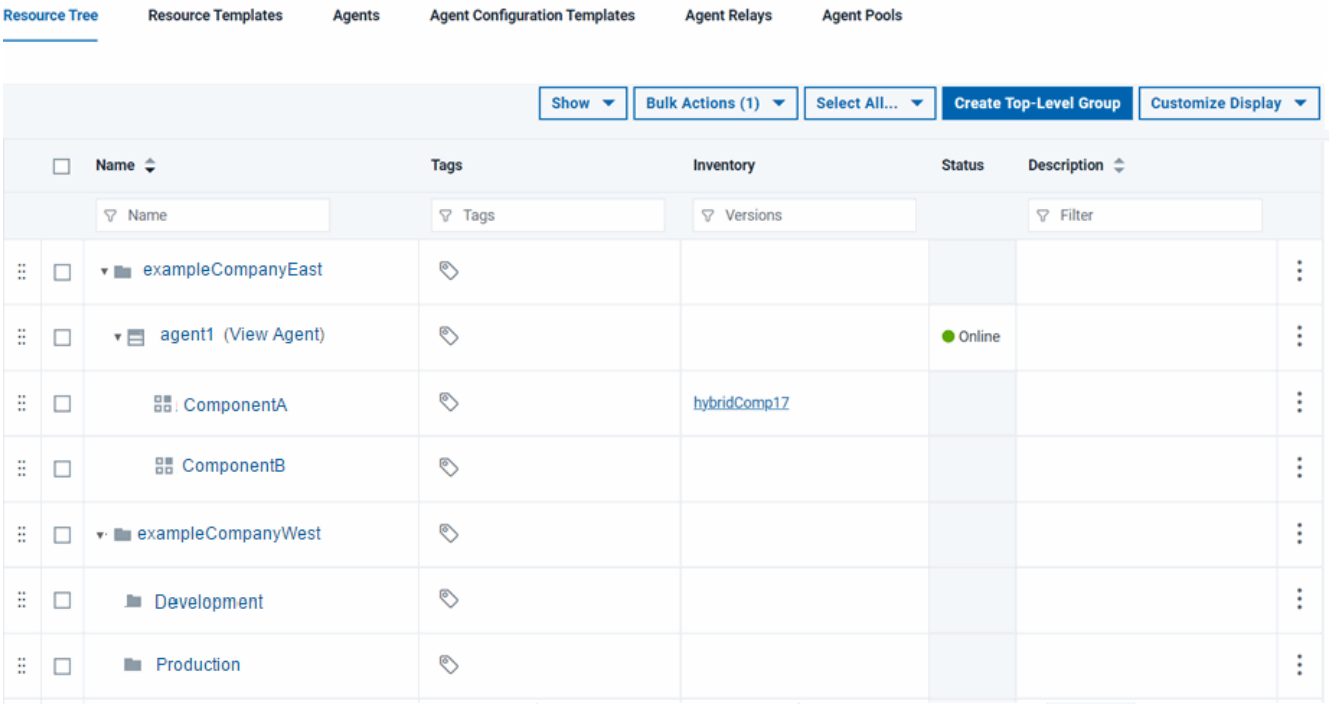
A *proxy resource* is a resource that is effected by an agent on a host other than the one where the resource is located. If an agent does not require direct interaction with the file system or with process management on the host, a proxy resource can be used. When a deployment must interact with a service that is exposed on the network (a database or Java™ Platform, Enterprise Edition server, for instance), the interaction can happen from any server that has access to the networked service.

A group-type resource can represent any logical collection of resources. In the following illustration, the top-level resource, `exampleCompanyEast`, represents a data center that contains the `Development` subresource (which represents an environment). In this instance, the `exampleCompanyEast` resource does not point to any higher level resource. The resource `ComponentA` represents a component, and delegates its automation up the chain until it reaches the `agent1` resource, which represents an agent. In order for an agent to be invoked, at least one resource must be defined for it.

Resources are created on the **Resources** page, which displays resources in a hierarchical tree structure. By displaying all resources across all environments, the **Resources** page provides a single place to see your entire infrastructure. Note: Resource names are limited to 255 characters.

The hierarchical structure is used wherever resources are used and not just on the **Resources** page, such as when assigning resources to environments. The hierarchical model is especially effective when several logical deployment targets reside on the same machine. In the accompanying illustration, the resources `ComponentA` and `ComponentB` could represent application servers on the same machine where the agent `agent1` is located. Both can be individually targeted for deployment.

Selecting a resource selects every resource below it, if any, in the hierarchy; for example, deleting or moving the `agent1` resource deletes or moves the `ComponentA` and `ComponentB` resources as well.



Whether you need one or multiple resources per environment is determined by your current infrastructure, deployment procedures, and other requirements. Many Deploy users have significant differences among environments. Whatever your structure, the **Resources** page displays the entire infrastructure across all environments.

If you are using IBM® WebSphere® Application Server, you can import information from cells as resources; see Importing resources from WebSphere Application Server on page 2325.

The resource tree

All of the resources on the Deploy server are organized into the *resource tree*. Top-level resource groups contain resources and other resource groups, and so on. Therefore, to refer to a resource, you must specify its path on the resource tree. For example, if an agent resource that is named `Agent1` is contained in the group `/Agents`, you can refer to the agent resource by the path `/Agents/Agent1`.

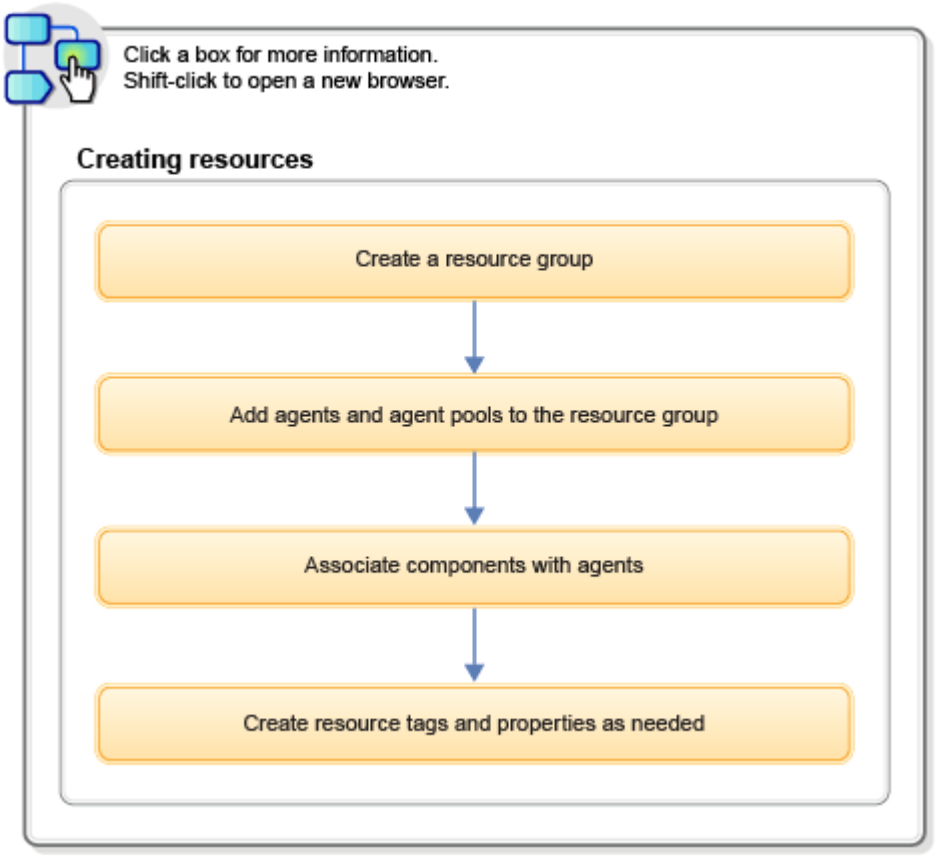
Resource groups

A *resource group* is a logical collection of resources. Resource groups enable collections of resources to be easily reused. Resource groups can manage multi-tenant scenarios, for example, in which several servers share resources.

Getting started with creating resources

Resources associate agents with components and, after they are mapped to an application environment, provide deployment targets.

The following diagram outlines the typical steps for creating and running a deployment:



1. Click this area for information about creating resource groups. on page 2315
2. Click this area for information about adding agents to resource groups. on page 2318
3. Click this area for information about creating component resources. on page 2317
4. Click this area for information about adding properties to resources. on page 2504

A resource group associates agents or agent pools with components. A resource group can itself belong to a parent group in a hierarchical collection. A component-type resource can point directly to the agent or agent pool that deploys it or point to another resource in a hierarchical chain of resources. If a resource is part of a hierarchy, it delegates its automation, if any, up the chain until it finds an agent.

Resources, just like the items they represent, can have properties tags.

Related topics:

- How to add tags to resources; see Adding tags to objects on page 2521.
- How to provide user credentials to resources; see Defining default user impersonation credentials on page 2332.

Modifying the resource tree

You can create, delete, and modify the contents of the resource tree.

Creating groups

Resource groups organize resources into logical groupings.

About this task

Resources are created on the **Resource Tree** page. The **Resource Tree** page displays resources in a hierarchical tree structure. Each row represents a resource and has a context-sensitive **Action** menu that is associated with it. The available actions depend on whether the resource represents an agent, agent pool, component, or organizational group.

In addition to the **Resource Tree** page, resources might be displayed in several ways, such as by adding resources to an environment.

1. Open the **Resource Tree** page.
2. Perform one of the following steps:

    a. To create a top-level group, click **Create Top-Level Group**.
    The top-level groups are ordered alphabetically.

    b. To create a resource group beneath an existing resource, display the **Action** menu for the resource, and then use the **Add Group** command.

    Result

    The **Create Resource** dialog box is displayed.
3. Enter a name and an optional description.
4. **Optional:** To specify criteria that are used to automatically populate the resource, select the **Include Agents Automatically** check box.

    Note: To automatically include agents, you must be a member of either the agent's team or the system team, such as the admin user. You must specify the team that will access the group, and that group must have access to the agent that you describe. You cannot assign the system team to the group. Existing agents matching the criteria are automatically included upon saving. The default interval for including new agents matching the criteria is every 5 minutes.

    Select **Name** or **Property**, and then select a filter criterion from the list. The available criteria are listed:
    - **Equals**
    - **Doesn't Equal**
    - **Contains**
    - **Doesn't Contain**
    - **Matches Regex** (match a regular expression)

    You can set more than one condition.
    If any agent-type resource in the filtered group satisfies the condition, it is preserved, otherwise it is removed. If agents that satisfy the condition exist, but they are not in the group, agent-type resources are created and added to the group.
5. **Optional:** If you added the group beneath an existing resource, perform one of the following steps:

a. If you want the team that manages the resource's parent to manage this resource, select the **Inherit Teams From Parent** check box.
The **Inherit Teams From Parent** check box is selected by default.

b. If you do not want the parent team to manage this resource, clear the **Inherit Teams From Parent** check box.
You can add a team by using the **Teams** action, then selecting a team.

6. **Optional:** If you added the resource below an agent, agent pool, or a component, the option to define default user impersonation credentials is available. See Defining default user impersonation credentials on page 2332 for information about setting impersonation credentials.
7. Save your work.

What to do next

For information about using resources in applications, see Mapping resources and components to environments on page 2363.

Creating agent-type and agent pool-type resources

Agent-type and agent pool-type resources represent the computers where components are deployed.

About this task

Resources are created on the **Resources** page. The **Resources** page displays resources in a hierarchical tree structure. Each row on the **Resources** page represents a resource and has a context-sensitive **Action** menu that is associated with it. The available actions depend on whether the resource represents an agent, agent pool, component, or an organizational group.

1. Open the **Resource Tree** page.
2. To create a resource, open the **Action** menu for the resource where you want to create the resource and click **Add Agent** or **Add Agent Pool**.
New resources are inserted below the selected resource. Agents and agent pools can be added below group-type resources.
3. From the Agent list, select the agent or agent pool that you want to associate with the resource.
4. In the **Name** field, enter a name for the resource.
By default, the resource name is the same as the agent or agent pool name.
5. Perform one of the following steps:

a. If you want the team that manages the resource's parent to manage this resource, select the **Inherit Teams From Parent** check box.
The **Inherit Teams From Parent** check box is selected by default.

b. If you do not want the parent team to manage this resource, clear the **Inherit Teams From Parent** check box.
You can add a team by using the **Teams** action, then selecting a team.

6. **Optional:** Define default user impersonation credentials.

For information about setting impersonation credentials, see Defining default user impersonation credentials on page 2332.

7. Save your work.

What to do next

For information about using resources in applications, see Mapping resources and components to environments on page 2363

Creating component-type resources

Component-type resources represent the components that are deployed to target environments.

About this task

Component-type resources are created on the **Resource Tree** page. The **Resource Tree** page displays resources in a hierarchical tree structure. Each row on the **Resource Tree** page represents a resource and has a context-sensitive **Action** menu that is associated with it. The available actions depend on whether the resource represents an agent, agent pool, component, or an organizational group.

1. Open the **Resource Tree** page.
2. To create a resource, open the **Action** menu for the resource where you want to create the resource and click **Add Component**.
   New resources are inserted below the selected resource. Components can be added below agents, agent pools, and other components.
3. From the **Component** list, select the component to associate with the resource.
4. In the **Name** field, enter a name for the resource.
   By default, the resource name is the same as the component name.
5. Perform one of the following steps:

   a. If you want the team that manages the resource's parent to manage this resource, select the **Inherit Teams From Parent** check box.
      The **Inherit Teams From Parent** check box is selected by default.

   b. If you do not want the parent team to manage this resource, clear the **Inherit Teams From Parent** check box.
      You can add a team by using the **Teams** action, then selecting a team.

6. **Optional:** If you added the component resource below an agent, agent pool, or another component, the option to define default user impersonation credentials is available. See Defining default user impersonation credentials on page 2332 for information about setting impersonation credentials.
7. If the component associated with the resource has any resource properties defined for it, they are displayed in the **Role Properties** area where you can set their value. If you change the default value, the **Reset to Default** action is displayed in the **Actions** field.
8. Save your work.

What to do next

For information about using resources in applications, see Mapping resources and components to environments on page 2363.

Adding agents to resource groups

You can find agents that are not associated with a resource and add them to a resource group.

1. Open the **Resource Tree** page.
2. Click **Show > Agents With No Resource**.
   Result
   The **Agents With No Resource** panel opens.
3. From the list of agents, select an agent by clicking the : : field in the first column of the list.
   The list is empty if all agents are associated with resources.
4. Drag the agent to the resource group to add the agent to that group.
5. Click **Hide Panel** to close the list of agents.

Moving and copying resources

Resource tree structures in the **Resources Tree** page can be configured by moving and copying the resources using the drag and drop feature.

About this task

Each row on the **Resource Tree** page represents a resource. Resources that are grouped under other resources are referred to as *subresources*. You can edit your organizational group nesting by moving and copying existing resources. Components and agents within a resource tree are also moved and copied along with the resource.

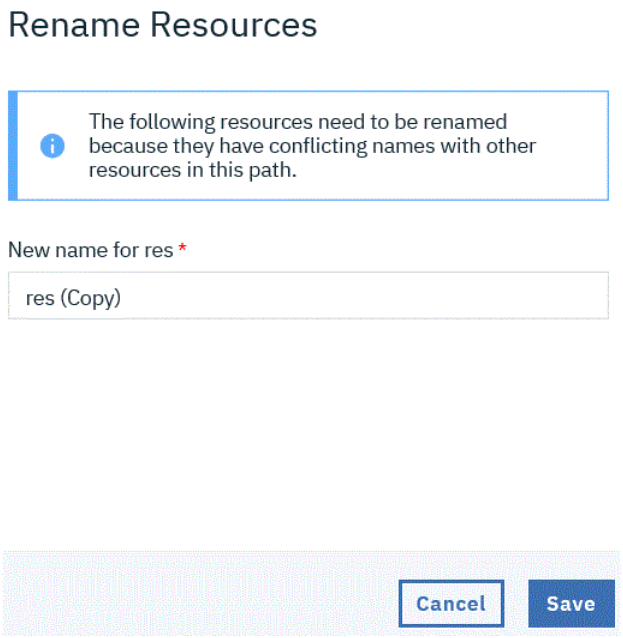
Procedure

1. Open the **Resource Tree** page.
2. Click and hold the mouse in the first column of the resource that you want to reposition.
   This column has small dots that are arranged in a box.
3. Move a resource by dragging the resource to a new location.
   Result
   While you are dragging the resource, a window appears to show the resource, as shown in the following figure:

   The left arrow sign (←) indicates that the row can be moved to the new location. The minus sign (–) indicates that row cannot be moved to the location.
4. Copy the resource by holding the Ctrl key while you drag it.
5. Position the pointer over the row where you want to copy the selected resource.

While holding the Crtl key, the plus sign (+) indicates the resource can be copied to the location, otherwise it shows minus sign.

6. Release the mouse button.
   The resource and any subresources that are associated with it are copied beneath the target row.
   When you copy a resource with a name that matches with an existing resource in the target location, the following window is displayed:



7. Save the resource with a new name.

Results

Results

The resources are moved or copied to the new location. Click the expand icon ▶ to expand the tree.

Deleting resources

When you no longer require the resources, you can remove them from the **Resource Tree** page and choose to delete them permanently.

Before you begin

You must be logged in as an administrator or must have the permissions to **Execute on Resources** and **Manage Children**.

About this task

A deleted resource is removed from the **Resource Tree**, but not from the system. The deleted resources are moved to the **Recycle Bin**. The deleted resources remain in the **Recycle Bin** for 14 days, after which they are deleted permanently from the system.

Notes:

- Resources that are part of resource template and dynamic resources, if deleted, are not moved to **Recycle Bin** and cannot be restored.
- If the deleted resource is mapped to an environment, the resource is deleted from the environment also. In case you still require the deleted resource, restore the resource, and map it to the environment again.
- In addition to the **Resource Tree**, resources might be displayed in several ways, such as by adding resources to an environment.

1. Login to HCL DevOps Deploy (Deploy).
2. Go to **Resources > Resource Tree**.
3. Select the checkbox that is associated with the resource to delete.

   Note: You can also select multiple resources that you want to delete.

4. From the **Actions** list, click **Delete**.
   The deleted resources are moved to the **Recycle Bin**.

   Note: The subresources that are associated with the selected resources will also be deleted.

Results

You have deleted the selected resources.

Related information

Deleting resources from the Recycle Bin on page 2320

Restoring the resources from the Recycle Bin on page 2321

Deleting resources from the Recycle Bin

When you no longer require the resources permanently, you can delete the resources from the **Recycle Bin** by setting up time and duration for the cleanup.

Before you begin

You must be logged in as an administrator or must have the permissions to **Execute on Resources** and **Manage Children**.

About this task

You can set up the time and duration for resources to get permanently deleted from the **Recycle Bin**.

Note: The deployment history cleanup process deletes all records in the database and the file system that are related to the deployments that you are deleting. The cleanup cannot be undone. Carefully choose the items that you delete.

1. Login to HCL DevOps Deploy (Deploy).
2. Go to **Settings > System Settings > Process History Cleanup** pane.
3. Turn on **Enable Deployment History Cleanup**.
4. Set up the time and duration in **Daily Cleanup Start Time** and **Daily Cleanup Duration** fields.

   The resources get permanently deleted according to the time and duration set by you.

Results

You have achieved to permanently delete the resources.

Related information

Cleaning up deployment history files on page 2133

Deleting resources on page 2319

Restoring the resources from the Recycle Bin on page 2321

Restoring the resources from the Recycle Bin

When needed, you can choose to restore the resources to their original locations in the **Resource Tree** after you delete them. When restored, the resources are restored as a child resource of the original parent resource. If the parent is moved, the resource is restored to the parent's new location. However, after resources are deleted from the **Recycle Bin**, they cannot be restored.

Before you begin

You must be logged in as an administrator or must have the permissions to **Execute on Resources** and **Manage Children**.

About this task

The **Recycle Bin** lists retained resources after you delete them. The recycle bin contains information such as the user who deleted the resource, the date, and the original path. You can sort and filter the resources by using this information and restore multiple retained resources at once.

Notes:

- You cannot restore a resource if the parent is also deleted. Therefore, you must restore the parent resource along with the child resource.
- Resources that are part of the resource template and dynamic resources, if deleted, are not moved to the **Recycle Bin**.

1. Login to HCL DevOps Deploy (Deploy).
2. Go to **Resources > Recycle Bin** tab.
3. Select the checkbox that is associated with the resource to restore.

   Note: You can also select multiple resources that you want to restore.

4. From the **Actions** list, click **Restore**.
   The resources are restored and available on **Resource Tree**.

Results

You have restored the resources.

Related information

Deleting resources on page 2319

Comparing and synchronizing resources

You can compare the contents of a resource group to another group or a resource template and then apply changes to the group or template.

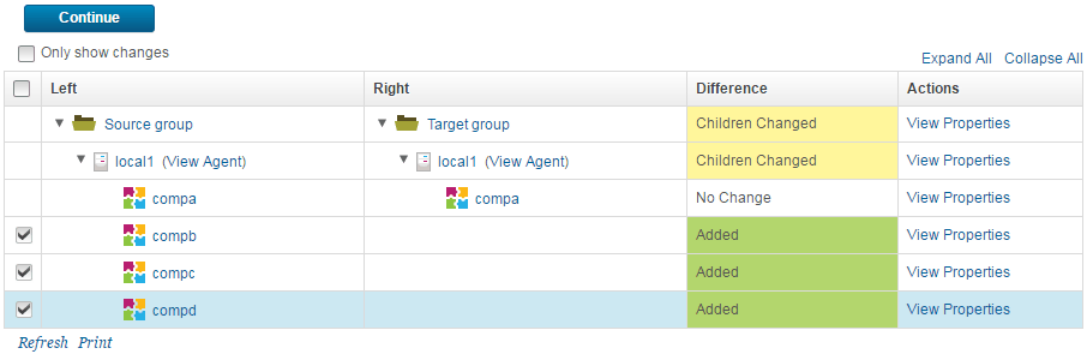
About this task

You can compare a group in the resource tree to another group or a resource template. You can apply changes, including properties and their values, from the first group that you select to the resource tree or second group.

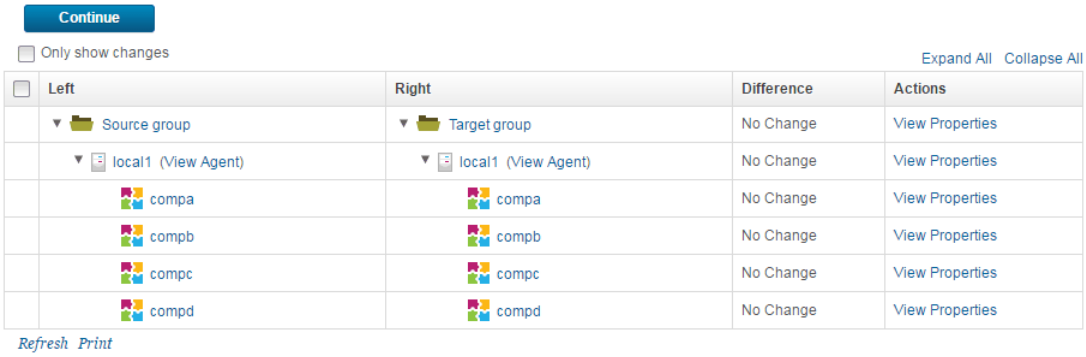
1. Open the **Resource Tree** page.
2. Display the **Action** menu for the folder that contains the source group and click **Compare or Synchronize**.
3. Compare the contents of a resource to either a resource template or another resource.
   Choose from:
   - To compare the contents of the resource to the contents of a resource template, take the following actions:
     a. From the **Compare Resources** window, select **Compare With Resource Template**.
     b. From the **Resource Template** list, select the name of the target resource template.
     c. Click **Save**.
   - To compare the contents of the resource to the contents of another resource, take the following actions:
     a. From the **Compare Resources** window, select **Compare With Resource**.
     b. From the **Compare Resources** list, click **Set**.

c. From the **Select Resource** window, select the target resource. You can expand resources and select a nested resource for the comparison.
d. Click **OK**.
e. Click **Save**.

4. On the **Resource Comparison** page, expand both resource trees.
5. Clear **Only show changes**.
6. Select each different resource from the left-hand, or source, resource to apply to the right-hand, or target, resource.
   The differences between the two resource trees are displayed. Each line of the resource tree can be one of the following states: **Children Changed**, **Added**, **Deleted**, or **No Change**.
   Example
   For example, the following image shows that the source resource tree contains three more agents than the target resource tree. All three additional agents will be added to the right-hand, or target, resource tree.



7. To review your changes, click **Continue**.
   Result
   The changes to the right-hand resource tree are displayed in the **Preview Changes** window.
8. Click **Apply Changes**.
   Result
   An **Alert** window confirms that the resource changes were applied successfully.
9. Click **Close**.
   Result
   The **Resource Comparison** page displays the updated status of the right-hand resource tree, as shown in the following image.



Since the contents of the groups are now identical, the statuses in the **Difference** column are all **No Change**.

Synchronizing resources from a template

You can add the contents of a resource template to the top-level resource group.

About this task

To add a resource hierarchy to a top-level group in the resource tree, synchronize the group to a resource template. You might complete this task to add a standard group of resources to an existing top-level group. The contents of the resource template are added directly to the group.

1. Open the **Resource Tree** page.
2. Display the **Action** menu for an empty group and click **Synchronize With Template**.
3. From the **Resource Template** list, select a resource template.
4. Click **Save**.

   Result

   The resource structure from the resource template is added to the group.

> **Note:** If you subsequently modify the resource template and then repeat the **Synchronize With Template** action, the modified resource structure is added to the group and the original resource structure is preserved. That is, the group will have two resource structures.

Adding resources from a template

You can add the structure from a resource template to a new group within an existing resource group.

About this task

To add a hierarchy of resource to a new group within a group in the resource tree, add resources from a template. You might complete this task to add a new node to an existing resource group.

1. Open the **Resource Tree** page.
2. Display the **Action** menu for the group and click **Add From Template**.
3. Provide a name for the new resource group.
4. From the **Resource Template** list, select a resource template.

5. Click **Save**.
   Result
   The resource structure from the resource template is added to the new group within the existing group.

Importing resources from WebSphere® Application Server

Use the WebSphere® Application Server - Deploy plug-in to import information about an IBM® WebSphere® Application Server system into a collection of resources. Typically, importing information by using the plug-in takes less time than manually creating resources to match the IBM® WebSphere® Application Server.

Before you begin

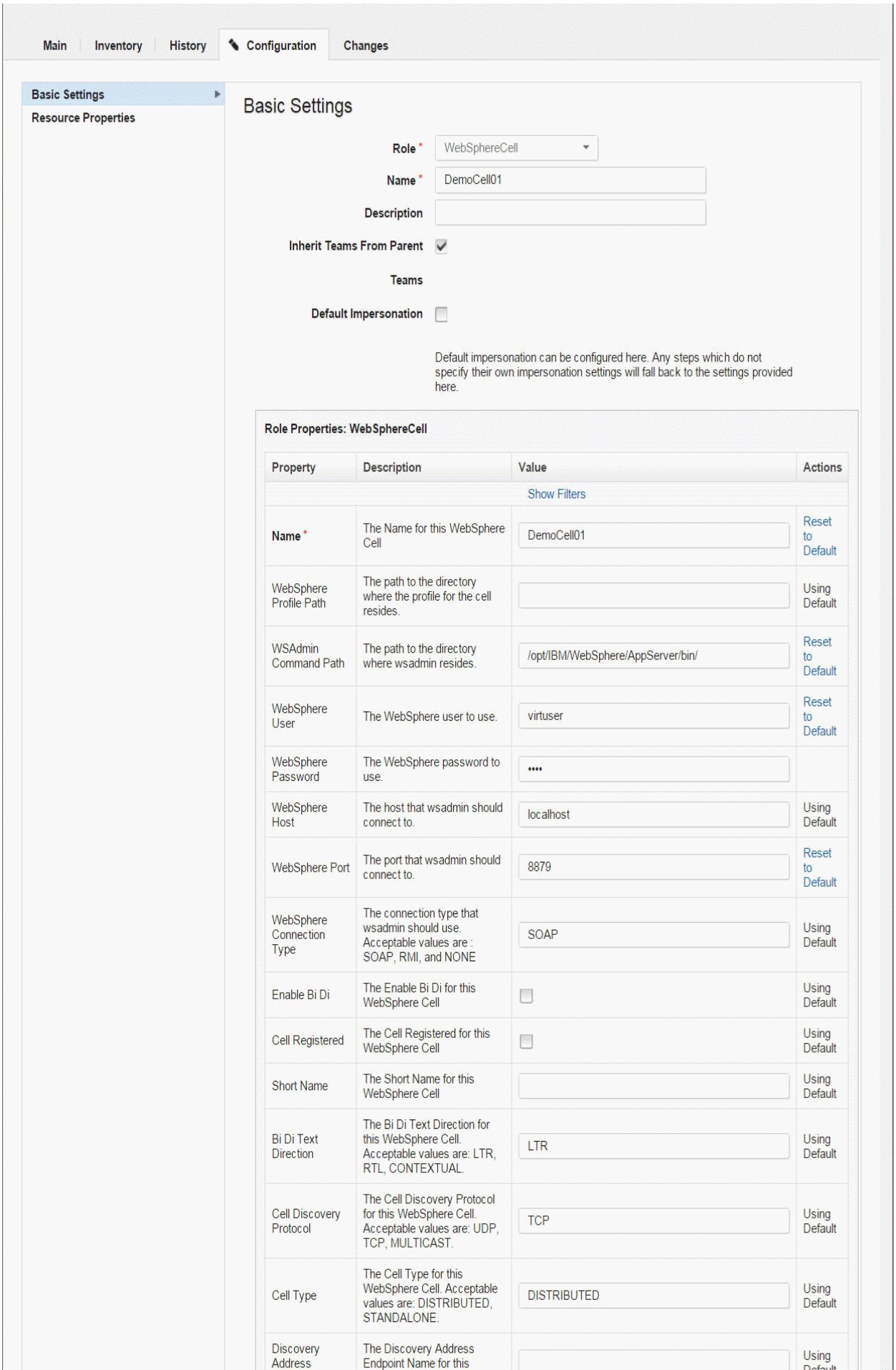
- Install the WebSphere® Application Server - Deploy plug-in. To download plug-ins, see the HCL™ Plug-ins website. Use the most recent version of the plug-in.
- Create a top-level resource group to contain agents. See Creating groups on page 2315.
- Install an agent on the WebSphere® Application Server system and add the agent as an agent resource under the top-level resource group.

About this task

Plug-ins can be developed and updated more frequently than Deploy provides release cycles. For the latest version of the WebSphere® Application Server - Deploy plug-in and the latest documentation, see the HCL ™ Plug-ins website.

1. On the Deploy server, click **Resources** to view the **Resource Tree** page.
2. If the `wsadmin` file is in a different location than one of the default locations, specify the file location as the value of the **wsadmin.path** property on the agent resource.

   The default location of the `wsadmin` file on Linux™ is `/opt/IBM/WebSphere/AppServer/bin/wsadmin.sh`. The default locations on Microsoft™ Windows™ are `C:\IBM\WebSphere\AppServer\bin\wsadmin.bat` and `C:\Program Files\IBM\WebSphere\AppServer\bin\wsadmin.bat`.
3. Click the **Actions** push button that is associated with the top-level group, and then click **Add Agent**.
4. In the **Create Resource** window, select the agent and then click **Save**.
   Result
   The new agent resource is displayed in the resource tree. Eventually, another resource is displayed under the agent resource. This resource is named `WebSphereCell` and represents the WebSphere® Application Server cell. You might need to click **Refresh** to see the cell resource, and it can take several minutes for the cell resource to be displayed.
5. Click the WebSphere® Application Server cell resource and then click **Configuration > Basic Settings**.
   The cell resource has the `WebSphereCell` role, which includes several properties that are related to the cell.
6. In the section that is labeled **Role Properties** for the `WebSphereCell` role, specify the connection information for the cell, including the locations of the profile and `wsadmin` file and the user name and password for the WebSphere® Application Server system, as in the following figure:


Main
Inventory
History
Configuration
Changes
Basic Settings
Resource Properties
Basic Settings
Role *
WebSphereCell
Name *
DemoCell01
Description
Inherit Teams From Parent
Teams
Default Impersonation
Default impersonation can be configured here. Any steps which do not specify their own impersonation settings will fall back to the settings provided here.
Role Properties: WebSphereCell
Property
Description
Value
Actions
Show Filters
Name *
The Name for this WebSphere Cell
DemoCell01
Reset to Default
WebSphere Profile Path
The path to the directory where the profile for the cell resides.
Using Default
WSAdmin Command Path
The path to the directory where wsadmin resides.
/opt/IBM/WebSphere/AppServer/bin/
Reset to Default
WebSphere User
The WebSphere user to use.
virtuser
Reset to Default
WebSphere Password
The WebSphere password to use.
WebSphere Host
The host that wsadmin should connect to.
localhost
Using Default
WebSphere Port
The port that wsadmin should connect to.
8879
Reset to Default
WebSphere Connection Type
The connection type that wsadmin should use. Acceptable values are : SOAP, RMI, and NONE
SOAP
Using Default
Enable Bi Di
The Enable Bi Di for this WebSphere Cell
Using Default
Cell Registered
The Cell Registered for this WebSphere Cell
Using Default
Short Name
The Short Name for this WebSphere Cell
Using Default
Bi Di Text Direction
The Bi Di Text Direction for this WebSphere Cell. Acceptable values are: LTR, RTL, CONTEXTUAL.
LTR
Using Default
Cell Discovery Protocol
The Cell Discovery Protocol for this WebSphere Cell. Acceptable values are: UDP, TCP, MULTICAST.
TCP
Using Default
Cell Type
The Cell Type for this WebSphere Cell. Acceptable values are: DISTRIBUTED, STANDALONE.
DISTRIBUTED
Using Default
Discovery Address
The Discovery Address Endpoint Name for this
Using

7. Click **Save**.
8. On the **Resources** page, click the **Actions** push button that is associated with the cell resource, and then click **Configure using WebSphere Topology Discovery**.

   Result

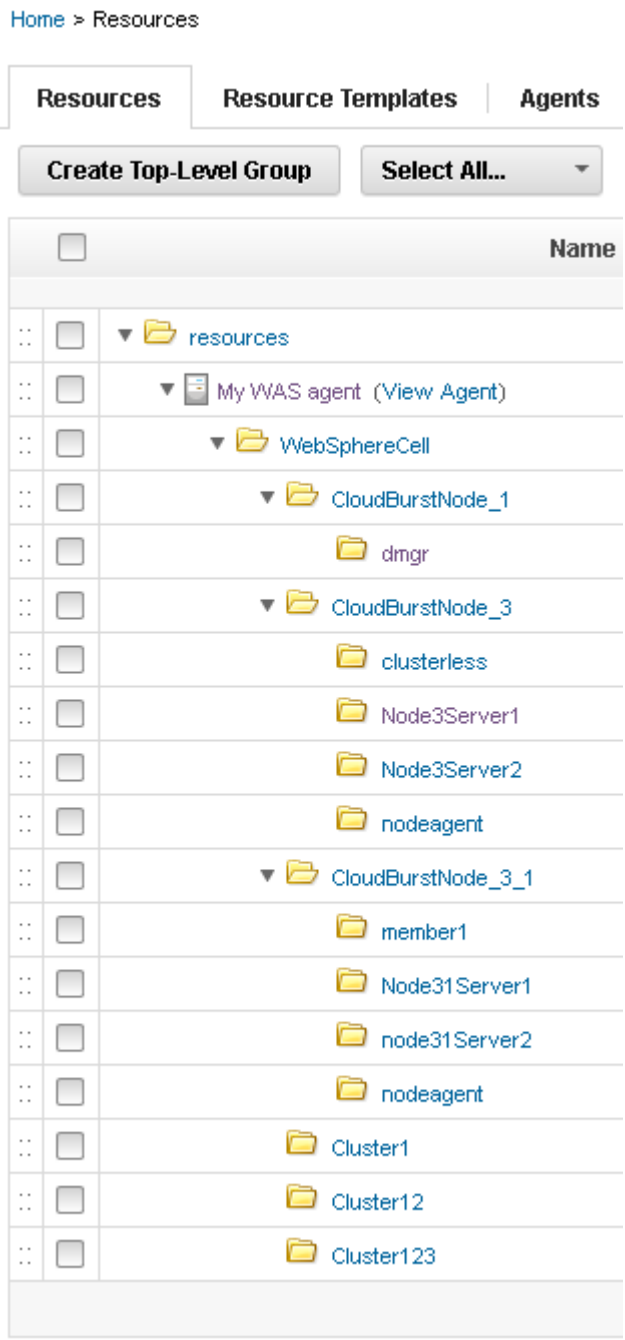
   The discovery step runs. This step might take several minutes. A log is displayed to show progress.
9. When the discovery process completes, close the log window, and then refresh the resource tree.

   Result

   Note: If you have multiple profiles defined for your WebSphere installation, you may discover all the profiles by using the `websphere.profilePath` property. See, https://hcl-software.com/plugins/666-2/

Results

Resources that represent the WebSphere® Application Server components are displayed in the resource tree. These new resources are displayed under the agent resource that you used to import the WebSphere® Application Server components. Each node, cluster, and server has a matching resource. The following figure shows resources that represent a WebSphere® Application Server system with three nodes:



What to do next

Add these resources to an environment by opening the **Resources** page for the environment and then clicking **Add Base Resources**.

Importing resources from WebSphere® Portal

With the WebSphere® Portal plug-in, you can discover resources for a portal cell and use plug-in steps to deploy artifacts.

Before you begin

- Create a top-level resource group to contain agents. See Creating groups on page 2315.
- Install an agent on the system that hosts the portal node.
- If IBM® WebSphere® Application Server is deployed in a directory other than the default directory, specify the **wsadmin.path** property on the agent.
    1. Click **Resources > Agents > *agent name* > Configuration > Agent Properties**.
    2. Add a property named **wsadmin.path**.
    3. Set the **wsadmin.path** property to the fully qualified path to the `wsadmin` script, including the script name.

       The default location of the `wsadmin` file on Linux™ is `/opt/IBM/WebSphere/AppServer/bin/wsadmin.sh`. The default locations on Microsoft™ Windows™ are `C:\IBM\WebSphere\AppServer\bin\wsadmin.bat` and `C:\Program Files\IBM\WebSphere\AppServer\bin\wsadmin.bat`.
- Install the WebSphere® Portal plug-in.
- Install the WebSphere® Application Server - Deploy plug-in.

About this task

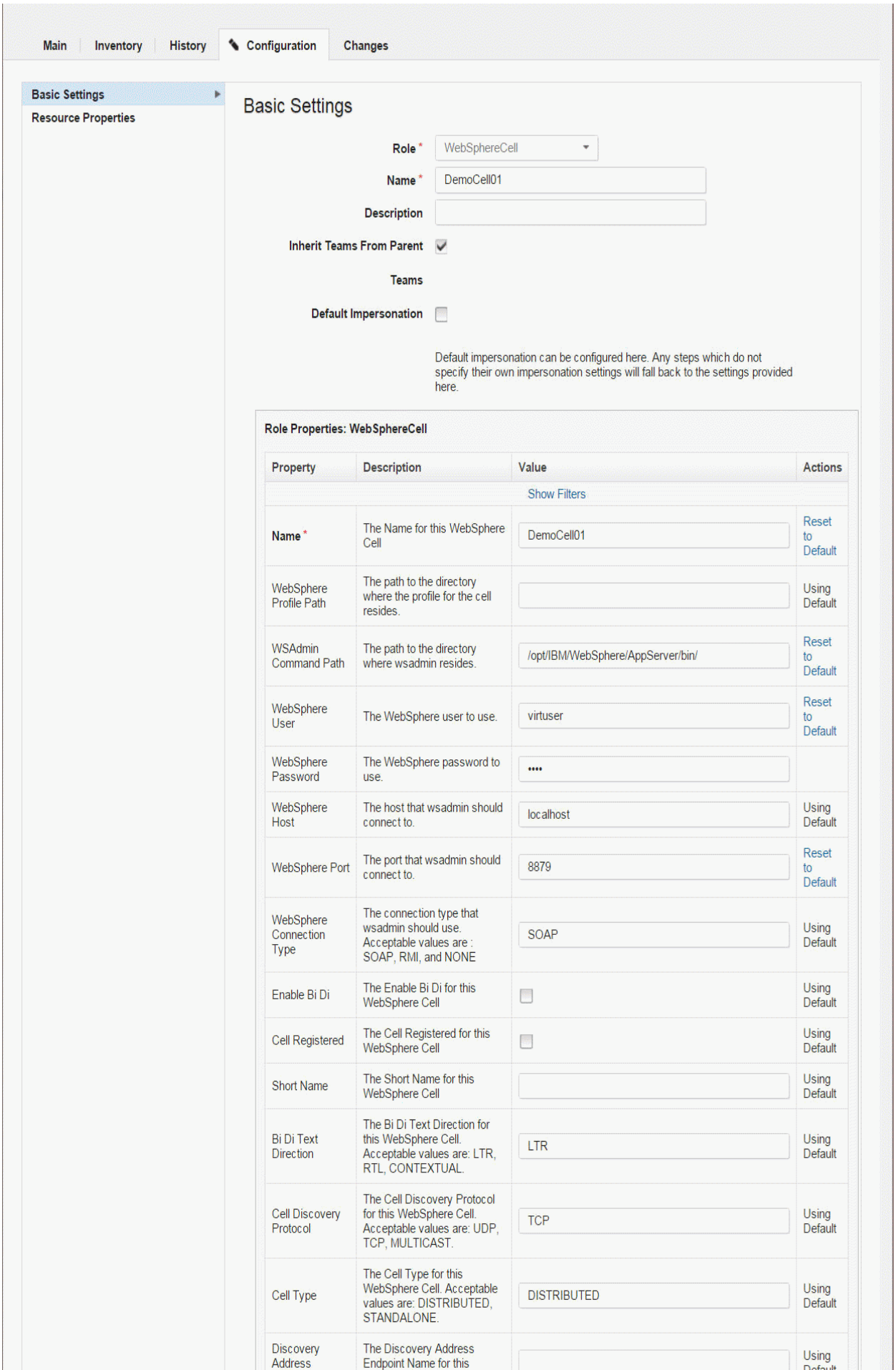
Plug-ins can be developed and updated more frequently than HCL DevOps Deploy (Deploy) provides release cycles. For the latest version of the WebSphere® Portal plug-in and the latest documentation, see the HCL Plug-ins site.

1. On the Deploy server, click **Resources** to view the **Resource Tree** page.
2. Click the **Actions** push button that is associated with the top-level group, and then click **Add Agent**.
3. In the **Create Resource** window, select the agent and then click **Save**.

   Result

   The new agent resource is displayed in the resource tree. Eventually, another resource is displayed under the agent resource. This resource is named `WebSphereCell` and represents the WebSphere® Application Server cell. You might need to click **Refresh** to see the cell resource, and it can take several minutes for the cell resource to be displayed.
4. Click the WebSphere® Application Server cell resource and then click **Configuration > Basic Settings**.

   The cell resource has the `WebSphereCell` role, which includes several properties that are related to the cell.
5. In the section that is labeled **Role Properties** for the `WebSphereCell` role, specify the connection information for the cell, including the locations of the profile and `wsadmin` file and the user name and password for the WebSphere® Application Server system, as in the following figure:


Main
Inventory
History
Configuration
Changes
Basic Settings
Resource Properties
Basic Settings
Role *
WebSphereCell
Name *
DemoCell01
Description
Inherit Teams From Parent
Teams
Default Impersonation
Default impersonation can be configured here. Any steps which do not specify their own impersonation settings will fall back to the settings provided here.
Role Properties: WebSphereCell
Property
Description
Value
Actions
Show Filters
Name *
The Name for this WebSphere Cell
DemoCell01
Reset to Default
WebSphere Profile Path
The path to the directory where the profile for the cell resides.
Using Default
WSAdmin Command Path
The path to the directory where wsadmin resides.
/opt/IBM/WebSphere/AppServer/bin/
Reset to Default
WebSphere User
The WebSphere user to use.
virtuser
Reset to Default
WebSphere Password
The WebSphere password to use.
WebSphere Host
The host that wsadmin should connect to.
localhost
Using Default
WebSphere Port
The port that wsadmin should connect to.
8879
Reset to Default
WebSphere Connection Type
The connection type that wsadmin should use. Acceptable values are : SOAP, RMI, and NONE
SOAP
Using Default
Enable Bi Di
The Enable Bi Di for this WebSphere Cell
Using Default
Cell Registered
The Cell Registered for this WebSphere Cell
Using Default
Short Name
The Short Name for this WebSphere Cell
Using Default
Bi Di Text Direction
The Bi Di Text Direction for this WebSphere Cell. Acceptable values are: LTR, RTL, CONTEXTUAL.
LTR
Using Default
Cell Discovery Protocol
The Cell Discovery Protocol for this WebSphere Cell. Acceptable values are: UDP, TCP, MULTICAST.
TCP
Using Default
Cell Type
The Cell Type for this WebSphere Cell. Acceptable values are: DISTRIBUTED, STANDALONE.
DISTRIBUTED
Using Default
Discovery Address
The Discovery Address Endpoint Name for this
Using

Note: Leave the **Cell Name** property blank.

6. Click **Save**.
7. On the **Resources** page, click the **Actions** push button that is associated with the cell resource, and then click **Configure using WebSphere Topology Discovery**.
   Result
   The discovery step runs. This step might take several minutes. A log is displayed to show progress.
8. Click **Save** to close the **Auto Configure** window.
9. Wait 30-60 seconds, and then click **Refresh**.
   A twistie is now displayed next to **WebSphereCell**. Expand the twistie and verify that the resource tree matches your WebSphere® Application Server topology. The resource tree contains deployable artifacts for WebSphere® Portal (theme, PAA, WAR, ConfigEngine command, and XMLAccess script).
10. On the **WebSphereCell** entry, click **Edit** and verify that **Cell Name** is correct.
11. Define properties for resources.
    Click the resource, then the **Configuration** tab, then **Resource Properties**.
    - For an ND topology, define the properties for Portal Cluster, Portal Primary Node, and Portal Secondary Node resources.
    - For a stand-alone topology, define the properties for the Portal Node resource.

    Define the following properties:
    - **portal.home**: Portal installation directory
    - **portal.profile.home**: Portal profile directory
    - **portal.admin.user**: Portal administrator user name
    - **portal.admin.password**: Portal administrator password
    - **portal.config.port**: Portal configuration port
    - **websphere.password**: WebSphere® administrator password
12. Create components for the resources.
    Create one component for each resource to deploy.
13. Add the components to the resources.
    a. Navigate to the resource.
    b. Click **Actions > Add Component**.
    c. Select the component to add to the resource.
14. Create the application.

    a. Click **Applications**.

    b. If you do not plan to use an existing application, click **Create Application**.
       You can also use an existing application that is attached to the cell to manage.

c. Enter an application name, and then click **Save**.

d. Click the **Components** tab, and then add the component that was created by the template generation procedure.

15. Create an application environment.

    a. On the **Environments** page, click **Create Environment**.

    b. Enter an environment name, and then click **Save**.

    c. Navigate to the environment editor.

    d. Click the environment name, and then click **Add Base Resource**.
    Select the group for your example cell and environment.

    e. Edit the environment properties for the component.
    Click **Configuration > Environment Properies**. In the **Component Environment Properties** list, enter values for each property shown. If you provided values previously, the values are already completed with default values.

16. Create an application process of the **Deployment** type for configuration and deployment.

    a. Click the **Processes** tab.

    b. Add a deployment step.

    c. Drag the **Install Component** step onto the process editor.
        i. Select the component that you created for the resource.
        ii. Choose the process for the component.

    Example
    For example, if you created a component to contain a Portal WAR file to deploy, select that component, and then choose the Deploy Portal WAR file process.

    d. Click **Save**.

    e. Save the process.

17. Run the process to test the newly created component.

Defining default user impersonation credentials

You can define default user impersonation credentials for a resource.

About this task

To access resources on a target system, you might need to access a specific user account on that system. Typically, the required credentials are defined in plug-in steps, but default credentials can be defined on resources. Steps that do not have their own credentials use credentials that are defined on associated resources.

1. Click **Resources**, and then click the resource's name.
2. Click the **Configuration** tab that is associated with the resource, not the **Configuration** tab at the top of the page.
   Result
   The resource's **Basic Settings** page is displayed.
3. Select the **Default Impersonation** check box.
4. In the **User** and **Password** fields, enter the target system user name and password that can access the resource.
   The password is encrypted as it is entered.
5. In the **Group** field, enter the authentication realm group.
6. To use the sudo command for impersonation, select the **Use Sudo** check box.
   If the target system uses a UNIX™ or Linux™ operating system, you can use this option. If the target system uses a Windows™ operating system, this option has no effect.
7. To always use the credentials that are defined on the resource and override any credentials that are defined in process steps, select the **Always Use Default Impersonation** check box.
8. Click **Save**.

Resource templates

A resource template is a model for a group of resources. The template contains a hierarchy of resources, groups, and agent prototypes that is a starting point for creating new resources.

Resource templates contain *agent prototypes*, which represent agents that are not yet installed or connected to an environment. When you create an environment, you can use a resource template as a model for the environment. Then, you can map real agents to the agent prototypes in the template.

Resource templates provide a pattern for an environment. For example, you can use resource templates to represent dynamic cloud environments. If you define environment templates in an application template, you must use resource templates to define the resource models for the environment templates. See Creating application templates on page 2369.

Note: If you create an environment with an environment template and then add component mappings to the resource template used by the template, the new component mappings are added to the environment. However, if you remove component mappings from the resource template, the component mappings are not removed from the environment.

Creating resource templates

To create a resource template, specify the contents and organization of the template, including resources, groups, and agent prototypes.

1. Click **Resources > Resource Templates** and then click **Create Resource Template**.
2. Specify a name for the template.

   Restriction: If you use this resource template with a blueprint that you provision to a cloud that uses virtual system patterns, the name cannot contain asterisk (*), backslash (\), or forward slash (/) characters.

3. **Optional:** In the **Teams** fields, specify the access information for the new resource template.

4. **Optional:** Check the box next to **Copy Teams to Root Resource** to make the root resource available to the same teams as the resource template.

   Note: This checkbox only affects creating resources from a resource template, not resources that are created by an environment template.

5. **Optional:** Under **Root Resource Teams** click the plus sign to add the teams that should be able to use the root resource of the resource template.
6. Click **Save**.
   Result
   The resource hierarchy for the new template opens in the editor. The hierarchy contains a top-level group.
7. Add resources to the template hierarchy by selecting the top-level group or another resource or group in the template, clicking **Actions**, and clicking a task, such as **Add Group**.
   The template can contain any number of resources.
8. **Optional:** Add agent prototypes to the resource template.
   An *agent prototype* is a resource that represents one or more agents that are not yet installed or connected to an environment.

   a. Next to a folder in the resource template, click **Actions (...) > Add Agent Prototype**.
      You can add only one agent prototype to each folder.

   b. In the **Create Resource** window, specify a description for the agent prototype.

   c. **Optional:** Specify a name pattern for the agents that the agent prototype represents.
      You might use an agent name pattern if you have a system for provisioning agents that is not cloud-based. When you create an environment from an application blueprint that is based on this resource template, you map agents that match the name pattern to the corresponding agent prototypes in the blueprint. For any agent prototypes that do not match existing agents, an entry is created in the environment for an agent with the expected name.

      Note: When you specify an agent name pattern, you can not use wildcards or regular expressions. You can use the following special properties as variables that represent the application name and the environment name: `${p:application.name}` and `${p:environment.name}`.

   d. Click **Save**.

9. **Optional:** Add components to the agent prototypes.

   a. Next to an agent prototype in the resource template, click **Actions > Add Component**.

   b. From the **Component** list, select a component that runs on the agent prototype.

   c. Enter a description.

   d. Define default user impersonation credentials.

For information about setting impersonation credentials, see Defining default user impersonation credentials on page 2332.

e. Click **Save**.

Components can be added to agents, agent pools, and agent prototypes.

Note:

- If you create an environment with an environment template and then add component mappings to the resource template, the new component mappings are added to the environment. However, if you remove component mappings from the resource template, the component mappings are not removed from the environment.
- Changes in a resource template structure are automatically picked up by the resource sync service but any change in the role properties of a resource template will not be synchronized.

10. **Optional:** Add component tags to the agent prototypes.

Note: If you use this resource template in an application template and assign component tags to the application template, add those component tags to agent prototypes.

a. Next to an agent prototype in the resource template, click **Actions > Add Component Tag**.

b. From the **Component Tag** list, select a tag for the component that runs on the agent prototype.

c. Enter a description.

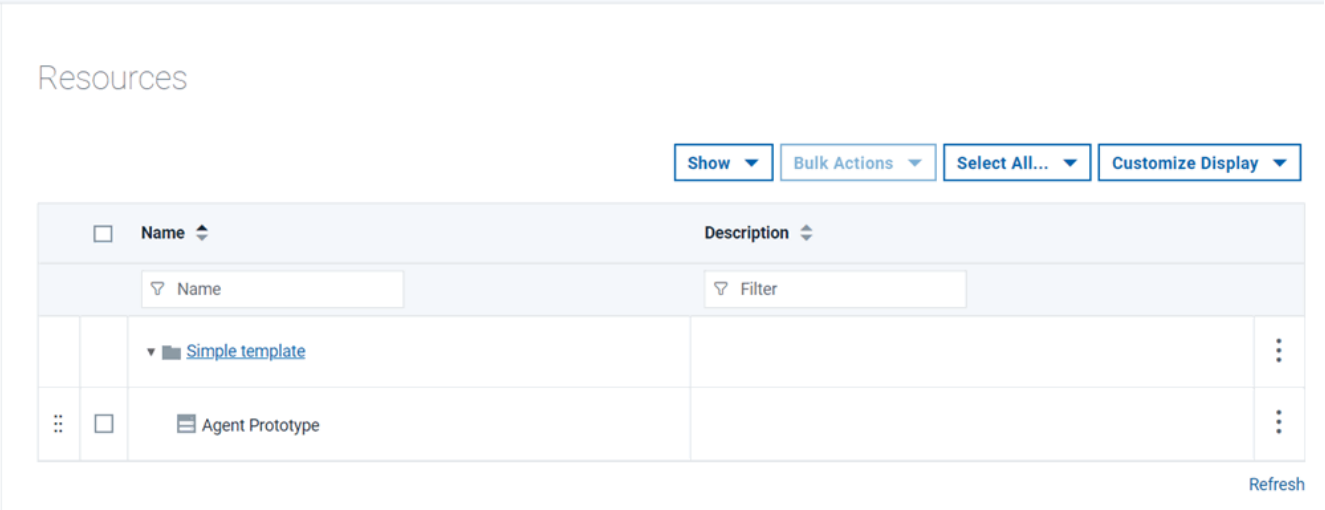
d. Define default user impersonation credentials.
For information about setting impersonation credentials, see Defining default user impersonation credentials on page 2332.

e. Click **Save**.

11. **Optional:** Add agent pool into resource template.
    a. Click the . . . icon next to the base resource of the resource template which appears on hover.
    b. Select **Add agent pool** option in the drop-down. Dialog will appear that lets you choose agent pool.
    c. Click **submit** and agent pool will be a child of the base resource.
    d. Select the . . . icon next to the new agent pool resource to **Add Component** and **Add Component Tag**.
12. **Optional:** Add agent into resource template.
    a. Click the . . . icon next to the base resource of the resource template which appears on hover.
    b. Select **Add agent** option in the drop-down. Dialog will appear that lets you choose agent.
    c. Click **Save** and agent will be a child of the base resource.
    d. Select the . . . icon next to the new agent resource to **Add Component** and **Add Component Tag**.

Example

For example, the following resource template contains a single agent prototype.



Note: Resource templates cannot be deleted if in use by an environment template, or if it's a parent of another resource template. When you delete a resource template all resources in that template will also be deleted.

Defining a new template from the resource tree

You can create a resource template from an existing resource group in the resource tree.

You might complete this task to reuse the hierarchy of a resource group in another location.

Note: If the group contains agents, the agents are included in the new resource template. The agents are not replaced with agent prototypes. Ensure that you use the new resource template appropriately.

1. Open the **Resource Tree** page.
2. Display the **Action** menu for a group, and then click **Define New Template**.
3. In the **Define New Template** window, set the resource template name.
4. **Optional:** Set the **Description**, **Parent Template**, and **Teams** for the resource template.
   See Creating resource templates on page 2333.
5. Click **Save**.
   Result
   The resource template is displayed.

Agents

An agent is a lightweight process that runs on a deployment-target host and communicates with the HCL DevOps Deploy (Deploy) server.

Agents are installed as Web agents on page 2096. Web agents use WebSocket connections and HTTPS for agent-server communication.

Agents do the actual work of deploying components and so relieve the server from the task, making large deployments that involve thousands of targets possible. Usually, an agent runs on the same host on which the resources it handles are located. A single agent can handle all the resources on its host. If a host has several resources, an agent process is started separately for each one. Depending on the number of hosts in an environment, a deployment might require many agents. For example, a test environment might contain a single web server, a single middleware server, and a single database server all running on the same host. A deployment to this environment might have one agent and three separate resources.

Agents have one or more matching *agent resources*, which represent the agents in the resource tree.

Agents are installed with the batch files that are provided with the installation files, see Installing agents from the command line. You can install agents on UNIX™ systems with the web application. Agents are run with the batch files that are included with the installation package.

Agents are unobtrusive and secure. Agent communications use SSL encryption and mutual key-based authentication for server-agent communication and employ HTTP, and HTTPS protocols to communicate with the server. See Agent security and communication on page 2095.

Agent configuration consists of assigning an agent to at least one environment; agents can be assigned to multiple environments. If an agent is assigned to several environments, it can do work on behalf of all of them.

Agent prototypes

An *agent prototype* is a resource that represents one or more agents that are not yet installed or connected to an environment. You can use agent prototypes in resource templates to represent resources that are not yet available. When you import a resource template from the cloud, an agent prototype is created for each node in the cloud pattern.

If you create a resource template instead of importing a resource template from the cloud, you can specify an **Agent Name Pattern** for any agent prototype that you add to the resource template. The agent name pattern can include the properties `${p:application.name}` and `${p:environment.name}`, but no other properties or wildcards.

Agent IDs

Each agent has a Universally Unique Identifier (UUID) that is created the first time you run an agent and it connects to the Deploy server. You can obtain an agent UUID from the **Agents** page by selecting an agent. The UUID is managed on the server. There is a one-to-one relationship between an agent name and its UUID. You can use either agent name or its UUID to delete an agent.

In addition to an agent's UUID, an agent has an endpoint ID that is generated as part of an agent installation process. An agent's endpoint ID is found in the `conf/agent/installed.properties` file of your agent installation directory with the property name **locked/agent.id**. This endpoint ID is broadcast to the server when the agent connects, and the server uses it to uniquely identify any given agent. When an agent is restarted and reconnects to the Deploy server, the Deploy server identifies the connecting agent by its endpoint ID and matches it with the agent UUID.

In REST API calls, APIs that require the agent's UUID or agent name, accepts the agent's endpoint ID as well.

Compact agents

Starting with V7.2.1, the agents are installed as compact agents.

A compact agent is a single-process mode for agents that enables a full upgrade of the agent software stack. It allows a Java upgrade, which was impossible without workarounds previously. In addition to this capability, compact agents also complete several other minor cleanups to the organization of the agent and its operation on Windows in particular.

Compact mode is supported on agents that run on Linux, Mac, and Windows platforms. On upgrade, the agents are switched to compact mode only on a supported platform.

Compact agent changes: Summary

A new installation or upgrade establishes the following changes:

• Running a compact agent doesn't show any monitor process.

• Restarting or upgrading changes the worker process identifier (PID).

• The `/monitor` directory is removed.

• The *JAVA_HOME* variable is set to the content of the *<DEPLOY_AGENT_HOME>*`/conf/agent/javahome.conf` file. The **set_env** setting is ignored.

• File changes in the agent installation directory are as follows:

File or folder	Previous location	New location	Description
shutdown.sh	/bin/shutdown	/var/shutdown	
restart.sh	/bin/restart	/var/restart	
classpath.conf	/bin/classpath.conf	/conf/agent/classpath.conf	Content is unmodified.
worker-args.conf	/bin/worker-args.conf	/conf/agent/javaopts.conf	The javaopts.conf file contains a list of options for the Java executable file that runs the agent.
upgrade.out	/bin/upgrade.out	/var/log/upgrade.log	
agent-upgrade	/agent-upgrade (or /bin/agent-upgrade	/var/upgrade/unpack	
air-agentupgrade.jar	/bin/air-agentupgrade.jar	/var/upgrade/air-agentupgrade.jar	
launcher.jar	/monitor/launcher.jar	/lib/launcher.jar	

The *<DEPLOY_AGENT_HOME>*`/conf/agent/javahome.conf` file contains a single line that is the path to the Java installation that runs the agent.

The following list identifies Windows-specific changes after a new installation or upgrade:

• The `/bin/agent.cmd` and `/native/<cpu>/winservice.exe` files replace the `/bin/agent.exe` file.

• The process removes the `/bin/run_agent.cmd`, `/bin/start_agent.cmd`, `/bin/stop_agent.cmd` files.

• If you run the agent as a service, restarting it restarts the entire service. Therefore, changes to the system environment are reflected in the agent's environment.

• The agent service configuration is no longer stored in the registry. Environment variables are defined in the `/bin/set_env.cmd` file and reflected in the agent's environment. This means that the service mode and `agent run` or `agent start` are more consistent.

• The process does not migrate the custom environment variables that are defined in previous service registry entries.

• Some side-effects of scripting in the `set_env.cmd` file might no longer apply. Changes to the environment are explicitly detected and saved. Other script continues to run and side-effects affects the entire system. For example, modifications to the file system still apply to the agent.

• The following files are replaced by the `/var/log/stdout` log file:

```
/var/log/stdout_XXXXXXXX.log
/var/log/stderr_XXXXXXXX.log
/var/log/service_XXXXXXXX.log
```

Note that the previous files are not removed but new files are not created.

• The service name cannot be customized. The service name is generated from a hash of the agent installation path and is unique to each installation on a computer.

Agent executable command-line commands

These are the command-lines that compact agent executable file accepts. The agent executables are the UNIX script or the Windows executable file.

Command	Description
run	Runs the agent from the console.
start [<pidfile>]	Runs the agent in the background. UNIX: The pidfile argument is optional but is typically used when it runs from an init script. Windows: The pidfile argument is not used.
stop	Stops the agent.

Command	Description
	It creates the "var/shutdown" file. The agent itself monitors for the existence of this file and terminates itself if it is detected. This means the user must have sufficient permission to write that file.
install-service	Windows-only command. This command installs the windows service. It must be run as an administrator with elevated privileges.
uninstall-service	Windows-only command. This command uninstalls the windows service. It must be run as an administrator with elevated privileges.
start-service	Windows-only command. This command starts the windows service. It must be run as an administrator with elevated privileges.
stop-service	Windows-only command. This command stops the windows service. It must be run as an administrator with elevated privileges. This command does not directly use the mechanism of the "stop" command. It requests the Windows service control manager (SCM) for the service to stop.

Agent metadata

The **Agent Metadata** tab provides additional information about the agent.

The **Agent Metadata** page contains agent information categorized into the following information types:

Category	Description
Java Homes	Lists all Java packages installed on the agent. Available attributes include name, vendor, version and status.
Configured Server URLs	Lists all server URLs configured with the agent. Server URLs can be used for proper assignment between the certificates and server hosts.
Configured Relay URLs	Lists all relay URLs configured with the agent.
Server Public Key Pins	Lists all server public keys pinned with the agent. Server key pin information is required to remove obsolete pins, and verify that the new pins are deployed.

Metadata	Lists metadata collected by web agents. An example agent metadata is shown below:
	
	`activeJavaHome C:\Program Files\Java\jdk-15.0.2`
	`verifyServerIdentityEnabled false`
	`serverRemotePort 51665`
	`serverRemoteIp 127.0.0.1`
	`serverPort 7919`
	`serverPinPerHostLimit 8`

Category	Description
	serverPinHostLimit 16 serverIp 127.0.0.1 serverId server-70F3qZScQmEku17HwjAT serverChannelId d53a0edc pluginListVersion 25 pid 2264 os WINDOWS javaHomeLimit 16 invocationTime 1710133017313 invocationId 77373ef3-cfff-46e7-8286-9f69127b542d hostName 10.134.119.207 hardwareFingerprint d098abdb42bd2607044660aaa0c82b204cfd hardwareAddress cpu X86_64 configuredJavaVersion 15.0.2+7-27 configuredJavaVendor Oracle Corporation configuredJavaHome C:\Program Files\Java\jdk-15.0.2 agentRemotePort 7919 agentRemoteIp 127.0.0.1 agentPort 51665 agentIp 127.0.0.1 agentHome C:\Program Files\devops-deploy\agent agentChannelId 0f6fcfca activeJavaVersion 15.0.2+7-27 activeJavaVendor Oracle Corporation

Agent relays

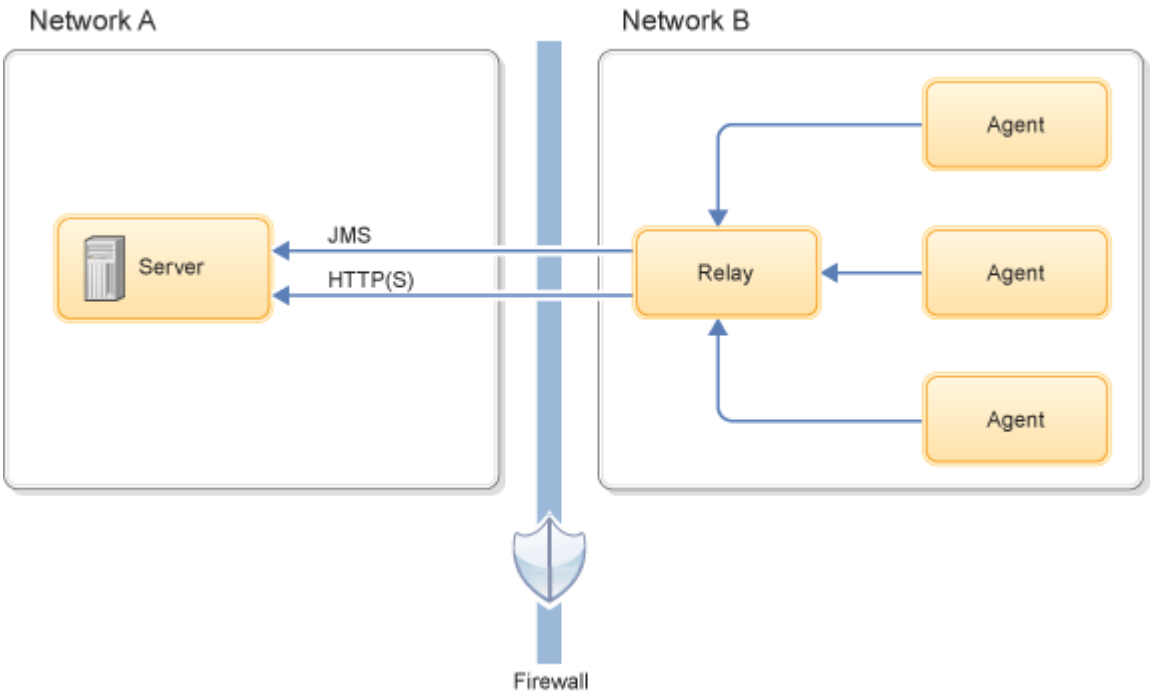
Agent relays coordinate communication between agents and the HCL DevOps Deploy (Deploy) server. Agent relays improve the performance of communication between the server and agents. Relays also allow agents to communicate through firewalls.

To enhance performance and reduce maintenance, Deploy can use *agent relays* to communicate with remote agents. Instead of each agent connecting directly to the server, agents connect to agent relays, which then connect directly to the server. Using agent relays in this way reduces load on the server because the server has fewer direct connections.

Agent relays also simplify communication across networks and firewalls. If multiple agents are in a remote network without an agent relay, each agent must connect to the server individually. In this case, each agent must have network permission to connect to the server, including firewall permissions. With an agent relay, the agents connect only to the relay, and the relay is the only system that contacts the server directly, such as with a VPN or tunnel.

Agent relays are often convenient when the agents are on cloud systems, because cloud systems are often separated from the Deploy server by a firewall or require a VPN or tunnel to reach the server.

The following figure is a simple diagram of communication between agents, an agent relay, and the server across a firewall. The agents connect directly to the agent relay, and the agent relay is the only system that connects to the server through the firewall.



The agent relay communicates with the server and the agents by using the same combination of HTTP, and HTTPS communication protocols that the agents use. As long as there is at least a low-bandwidth WAN connection between the server and agent relay, the Deploy server can send work to agents in other locations.

Note: Starting with version 7.0.5. the agent relay uses a new proxy by default. Only HTTPS traffic can be proxied with this new proxy, but the relay has historically used HTTPS. If for any reason the relay needs to be able to proxy HTTP traffic then the old proxy can be enabled by including the following line in the relays `agentrelay.properties` file: `agentrelay.http_proxy.use_rabbit=true` The new proxy can now support up to 10,000 connected web agents, although to reach maximum scalability, OS file descriptor and ephemeral port limits may need to be raised.

Agent relays open the connections to the Deploy server; the server never opens connections to the agent relays. Agents open the connections to the agent relay; the agent relay never opens connections to the agent. Agent to server communication is always encrypted. HTTP 20080 port is used to establish a tunnel between agent and server to transfer encrypted data. By default, agent relays open connections on the following ports for communication:

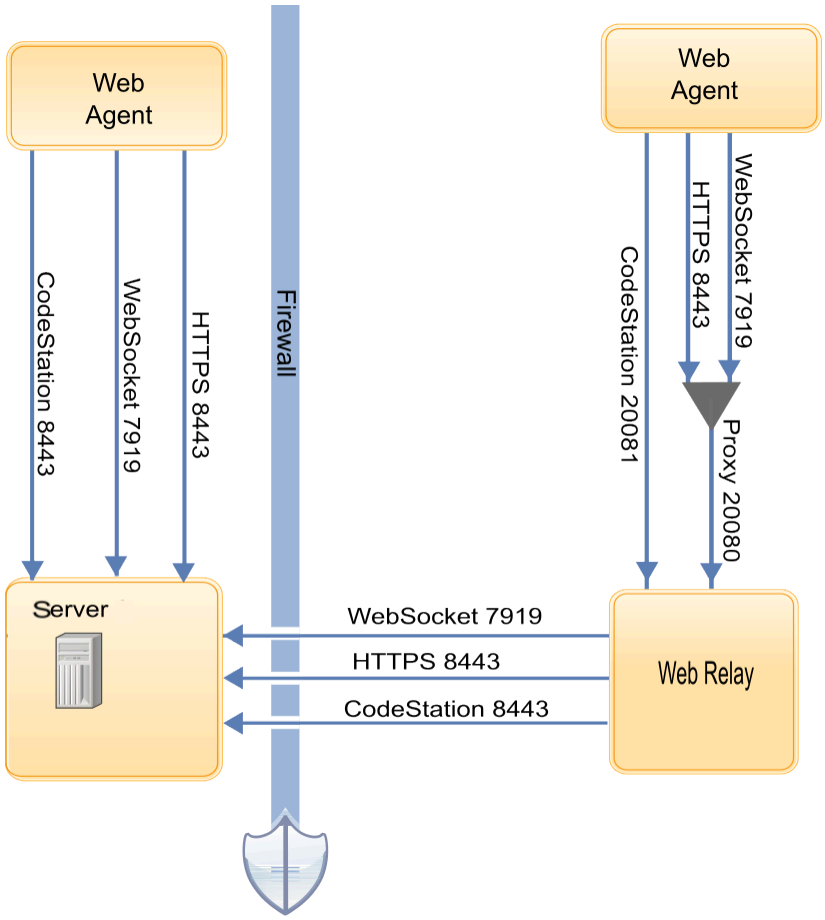
- 8080 and 8443 for HTTP and HTTPS connections to the server

Agent relays accept connections on the following default ports:

- 20080 for HTTP proxy connections from agents.
- 20081, the encrypted port, for artifact cache requests from agents.

The server does not open connections to agents or agent relays.

The following diagram shows the main default ports that are involved in communication between agents, agent relays, and the server.



You can also cache artifacts on agent relays to improve system performance. See Caching artifacts on agent relays on page 2125.

Example

Example

The following simple artifact move illustrates the mechanics of remote communications:

1. A remote agent starts and establishes a connection to the agent relay through WebSocket
2. The agent relay alerts the Deploy server through WebSocket that the remote agent is online.
3. The server sends an artifact download command to the relay through WebSocket, and the relay delivers the message to the remote agent, also through WebSocket.
4. The server locates the artifacts. If the artifacts are not in the relay's CodeStation cache, the server sends the artifacts to the relay over HTTP or HTTPS.
5. The agent relay streams the artifacts directly to the agent over the server-relay HTTP or HTTPS connection.
6. After the remote agent completes the work, it informs the server through the agent relay WebSocket connection.

Applications

Applications are responsible for bringing together all the components that must be deployed together.

Applications do this task by defining the different versions of each component and defining the different environments that the components must go through on the way to production. In addition, applications also map the constituent hosts and systems (called resources) that a component needs within every environment.

Applications also implement automated deployments, rollbacks, and similar tasks. These tasks are called processes. However, at the application level, processes are concerned only with the components and resources that are necessary for deployment, and related tasks. By contrast, component processes are concerned with running commands and related tasks.).

Applications also introduce snapshots to manage the different versions of each component. A *snapshot* represents the current state of an application in the environment. Typically, the snapshot is generated in an environment that has no approval gates. This type of environment is called an uncontrolled environment. For most users, the snapshot is pushed through the pipeline.

Note: Before you configure an application, install at least one agent in a target environment. For evaluation purposes, the agent can be on the same system as the server. In addition, you must also add at least one resource group to the agent. See Resources on page 2312.

Environments

An environment is a collection of resources that host the application. Environments typically include host systems and HCL DevOps Deploy (Deploy) agents. When a deployment is run, it is always done so in an environment. While environments are collections of resources, resources can vary per environment.

For example, environment 1 might include a single web server, a single middleware server, and a single database server. Deploy represents these systems as three separate resources that are running in environment 1. Environment 2, however, might include several clustered databases and servers. To deploy the same components on these different environments, Deploy organizes the resource elements with resource groups. The server also keeps an inventory of everything that is deployed to each environment. In this way, the Deploy server manages the contents of each environment and tracks the differences between those environments.

Processes

A *process* plays a coordination role. They are authored with a visual drag-and-drop editor, and composed of steps that call the component processes. For example, to deploy the application you might run a process that is called Deploy. This Deploy process would in turn run the requisite components and run the deployment.

Snapshots

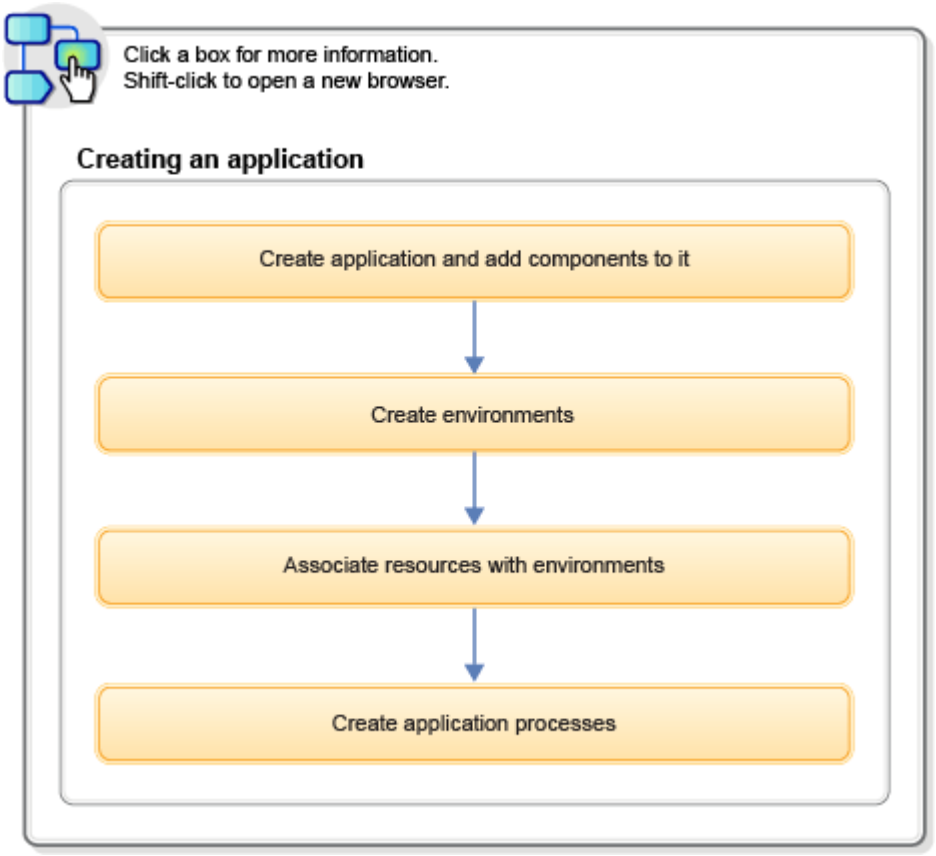
Snapshots specify what combination of component versions you deploy together. They are models that you create before you deploy the application. A snapshot specifies the exact version for each component in the application. When a snapshot is created, Deploy gathers together information about the application, including the component versions, for an environment. Typically, the snapshot is generated in an environment that has no approval gates. This kind of environment is called an

uncontrolled environment. For most users, the snapshot is pushed through the pipeline. Typically, one of the environments remains uncontrolled to allow for snapshots. When a successful deployment runs in the uncontrolled environment, a snapshot is created based on the state of the application within the environment: thus capturing the different versions of the components at that time. As the application moves through various testing environments, for example, Deploy ensures that the exact versions (bit for bit) are used in every environment. After all the appropriate stages and approvals for a snapshot are complete, the snapshot is pushed to production.

Getting started with creating applications

Applications associate resources with environments and define processes to run deployments.

The following diagram summarizes the steps for creating applications:



1. Click this area for information about creating applications. on page 2346
2. Click this area for information about creating application environments. on page 2354
3. Click this area for information about mapping resources to environments. on page 2363
4. Click this area for information about creating application processes. on page 2428

After defining the application itself, you identify the components that it manages, and create application processes that coordinate their deployment. Finally, you create the environments that serve as deployment targets, and map resources to them.

Related topics:

- About HCL DevOps Deploy (Deploy) properties; see Properties on page 2500
- How to create snapshots; see Snapshots on page 2383
- How to import applications; see Importing applications on page 2347

Creating applications

You can create an application from scratch or import an existing one.

Before you begin

Before you configure an application, ensure that at least one agent is installed in a target environment. For evaluation purposes, the agent can be on the same system as the server. See Resources on page 2312.

Some of the configuration options might be restricted based on the role assigned to you. Check with your administrator, if you need a specific option enabled.

About this task

See Importing applications on page 2347 for information about importing applications.

1. Display the **Create Application** dialog box **Applications > Create Application**, and enter the following information:
   - Typically the **Name** and **Description** correspond to the application you plan on deploying.

     Restriction: If you deploy this application to a cloud that uses virtual system patterns, the application name cannot contain asterisk (*), backslash (\), or forward slash (/) characters.

   - To grant teams access to the application and set its security type, click the **plus sign** icon by **Teams**, select values from the **Team** and **Type** lists, and click **Add**.
   - To create an application from a template, from the **Application Template** list, select the application template name. Then, from the **Application Template Version** list, select either the application template version number or to ensure that the application always meets the latest requirements of the template, **Always Use Latest**. See Creating applications from application templates on page 2371.
   - **Notification Scheme**. HCL DevOps Deploy (Deploy) includes integrations with LDAP and email servers that enable it to send out *notifications that are based* on events. For example, the default notification scheme sends out an email when an application deployment fails or succeeds. Notifications also play a role in approving deployments: Deploy can be configured to send out an email to either a single individual or to a group or people (based on their security role) notifying them that they must approve a requested deployment. See Creating Notifications in a Notification Scheme on page 2081.
   - If you want the application to require that every component is versioned, click the **Enforce Complete Snapshots** check box.
   - If you want the application to only deploy changed versions by default, click the **Only Changed Versions Default** check box.
2. Click **Save**.

   Result

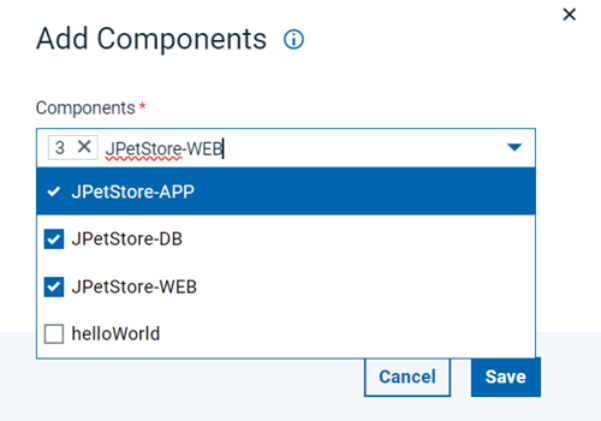
The new application opens.

3. Add one or more components to the application:

    a. Within the application, click the **Components** tab, which is not to be confused with the global **Components** tab.

    b. Click **Add Component**.

    c. In the **Add a Component** window, expand the **Select a Component** list and select the check boxes next to one or more components, as shown in the following figure.
    Example



    d. Click **Save**.

What to do next

After you create an application, you:

- Create an environment (Creating environments on page 2354)
- Associate an agent with the environment (Mapping resources and components to environments on page 2363)
- Create an application process (Application processes on page 2428)

Importing applications

If you exported an application, you can import the application on a different server. When you import an application, you can create a new application or upgrade an existing application.

Before you begin

Ensure that you have exported an application from the source server and copied the JSON or YAML file to the destination server. See Exporting applications on page 2352.

Additionally, you must know that:

- To share applications that have secured properties among Deploy servers, you exchange keys from each server's encryption keystore. See Sharing secured properties among servers on page 2350.
- The version of the destination server must be the same as of the source server.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

About this task

To move applications between servers, you export the applications and import them to a different server. When you import an application, components and other items that are associated with the application are also imported, including their properties and processes (if they are available to the importing server).

Note: Notification schemes are not imported.

For information about templates that are associated with the imported components, see Importing components on page 2296.

Notes:

- If imported components have the **Import Versions Automatically** parameter set to true, Deploy automatically imports component versions if the artifacts are accessible to the importing server.
- You might receive an error message that indicates the Java™ heap space is out of memory when you attempt to import an application. To resolve the issue, increase the Java™ heap size. Open the `server_directory/bin/set_env` file, and set the Java™ heap size to a higher value, such as 2 GB.

1. Open the **Import Applications** dialog box by clicking **Applications > Import Applications**.
2. To upgrade an existing application, enable the **Upgrade Application** option. To create a new application, leave **Upgrade Application** cleared.
   If the name of the application in the JSON file (not the name of the file itself) matches an existing application name, then the application parameters are updated. If items such as processes or properties exist, new versions are created. New items, such as processes, environments, components, or properties, are also added. If the name of the application is not found, the import action has no effect.

   Note: The application name is the first parameter in the defined file. This example shows the first parameter in the respective file:

   ```
   "name": "helloWorldApplication",
   ```

3. To import the application with snapshots, enable the **Import with Snapshots** option.
4. To import the full base resource trees mapped to the application environments, enable the **Import Full Base Resource Trees** option.

   When selected, the resource mapping of the imported application environment matches with existing resources on the source server.

Note: The resources must exist on the source server for the application environment to map to.

5. Configure how to import the items that are associated with the application.
   When you import an application, all application templates, components, generic processes, and resource templates that are associated with the application are also imported. For each item type, you must select the option from the list that determines how each item type is imported or upgraded. If your application does not contain items of a type, accept the default value.

   Application Template Upgrade Type

   If the application was created from an application template, you must specify how to import the application template.

   Component Upgrade Type

   If the application contains components, you must specify how to import them.

   Note: The components must be on the importing server.

   Process Upgrade Type

   If the application processes contain generic processes, you must specify how to import them. If you are also upgrading components, then the generic processes that are called in the component processes are also affected. The import process has different results if the generic processes exist or not.

   Note: The server tracks generic processes by name. If a generic process on the server is assigned the same name as a generic process that is imported, then the generic process exists on the server.

   Resource Template Upgrade Type

   The application template might also contain resource templates. Specify how to import resource templates that have the same names as existing resource templates.

The following table lists the options available for the above upgrade types:

Table 730. Upgrade options that are available for each item type

Option	Description
Use Existing *Item Type*	To use the same item that is used by the imported application, select this option. The new application contains references to the item. If you are upgrading, the application uses the imported item.
Create *Item Type*	To create an item that is based on the item that the imported application uses, select this option. A new item is created that is based on the item of the imported application. If you are upgrading, the application uses the newly created item, and it no longer uses the item that it previously used.

Option	Description
Fail If *Item Type* Exists	When you want to create a fresh installation, select this option. If you are creating an application, Deploy creates both a new application and item unless the item exists, in which case the application is not imported. If you are upgrading, the upgrade fails if the item exists on the importing server.
Fail If *Item Type* Does Not Exist	To ensure that an item is on the importing server, select this option. If you are creating an application, Deploy creates both a new application and item unless the item does not exist, in which case the application is not imported. If you are upgrading, the upgrade fails if the imported item does not exist on the importing server.
Upgrade *Item Type* If Exists	To upgrade the existing item, select this option. This option creates an application and upgrades the existing item with data from the imported application. If you are upgrading and an existing item matches an imported one, the item is upgraded. If the imported item matches the existing one, the imported item is used.

6. Click **Choose File**, and then select the name of the JSON or YAML file that contains the application definition.

   Note: You must create the JSON or YAML file that contains the application definition on a server that uses the same version of Deploy as the destination server.

7. Click **Submit**.

Sharing secured properties among servers

To share applications that have secured properties among HCL DevOps Deploy (Deploy) servers, exchange keys from each server's encryption keystore.

Before you begin

Make sure that the location of the Java™ keytool command is on the **PATH** variable of the servers. The `keytool` application is included in the Java™ developer kit and is not part of Deploy.

About this task

To share applications that have secured properties, export the server's key and import it into each target server. This process enables applications from the exporting server to be used by the other servers. Repeat the process for each server with applications that you want to share.

1. On the first server, open a command-line window, and go to the server installation `conf/server` directory.
2. Find the value of the **encryption.keystore.alias** property in the *`install`*`/conf/server/installed.properties` file.

   Example

For example, the following code shows a value of `abcdkey1234`:

```
encryption.keystore=../app_data/conf/encryption.keystore
encryption.keystore.alias=abcdkey1234
```

You must have this alias to complete the next steps.

3. Go to the folder that contains the keystore file `encryption.keystore`.
   The default location is *appData*`/conf/encryption.keystore`, where *appData* is the application data folder.
4. Run the following command to import the server key into a temporary keystore.
   You must enter this command on a single line.

```
keytool -importkeystore -srckeystore encryption.keystore
  -srcstorepass srcPassword
  -srcstoretype pkcs12
  -alias alias
  -destkeystore temp.keystore
  -deststorepass tempPassword
  -deststoretype pkcs12
```

   - For the *srcPassword* variable, specify the password for the server keystore. The default password is `changeit`.
   - For the *alias* variable, specify the value of the **encryption.keystore.alias** property.
   - For the *tempPassword* variable, specify a password for the temporary keystore. You will use this password later.
5. Copy the temporary keystore, which is named `temp.keystore` in the previous example and store it in the appropriate folder on the second server.
   Check the **server.appdata.dir** property in the `installed.properties` file to determine where to store the keystore.
   - If the **server.appdata.dir** property has a value of two periods (..), copy the keystore to the *install*`/conf/` folder on the second server.
   - If the property has a value other than two periods, the server uses an application data folder. Copy the keystore to the *appdata*`/conf/` folder on the second server.
6. On the second server, open a command-line window, and go to the server installation *install*`/conf/server` directory.
7. Run the following command to import the key in the temporary keystore into the server keystore.
   You must enter this command on a single line.

```
keytool -importkeystore -srckeystore temp.keystore
  -srcstorepass tempPassword
  -srcstoretype pkcs12
  -alias alias
  -destkeystore encryption.keystore
  -deststorepass destPassword
  -deststoretype pkcs12
```

   - For the *tempPassword* variable, specify the password for the temporary keystore.
   - For the *alias* variable, specify the **encryption.keystore.alias** property of the first server, not the current server.
   - For the *destPassword* password, specify the password for the current server keystore.
8. Restart the second server.
9. **Optional:** Delete the temporary keystore file from each server.

Results

The encryption key from the first server keystore is now on the second server keystore. As a result, the second server can now decrypt applications from the first server.

You can verify that the second server keystore has the key by running the following command on the second server:

```
keytool -list -keystore encryption.keystore
  -storepass password
  -storetype pkcs12
```

For the *password* variable, use the password of the server keystore.

This command lists the keys in the server keystore. If you copied the keys successfully, the list includes at least two keys: one from each of the servers. These keys are listed according to the **encryption.keystore.alias** properties of the respective servers. For example, the following output shows two keys:

```
Your keystore contains 2 entries

efghkey5678, Oct 15, 2013, SecretKeyEntry,
abcdkey1234, Nov 22, 2013, SecretKeyEntry,
```

What to do next

Repeat this process to copy keys between other servers.

Exporting applications

Exporting applications creates a JSON file (file extension `json`) that contains the properties, components (and their associated properties and processes), and processes of the applications.

About this task

For information about JSON, see http://www.json.org/.

Note: Exporting applications in this way includes only the most recent version of each component process and application process.

When you export an application, components, snapshots and other items that are associated with the application are also exported, including their properties and processes (if they are available to the exporting server). Notification schemes are not exported.

To export one or more applications:

1. On the **Applications** pane (**Applications**), select the check box next to one or more applications.
2. At the top of the table, click **Actions** and then click **Export**.

   Result

   The file is saved in the default download location for your browser.

What to do next

You can import the application to another server that uses the same version of HCL DevOps Deploy (Deploy). See Importing applications on page 2347.

Rational Team Concert™ Build

The Rational Team Concert™ Build system defines resources that are used to describe and manage builds.

The Rational Team Concert™ Build system in IBM® Rational® Team Concert™ defines the following types of logical resources that are used to describe and manage builds:

Build definition

A build definition describes:

- The workspace to process.
- The Ant scripts and targets to run against the workspace.
- The schedule for the build.
- The properties to simplify the configuration of the build.
- The build engines that can handle build requests for the build definition.

Build engine

A build engine represents a Build System Toolkit Jazz™ Build Engine running on a designated build computer. A Jazz™ Build Engine running on a build computer correlates itself to a build engine in Rational® Team Concert™ by specifying the unique identifier of the build engine.

Build request

A build request represents a scheduled or explicitly issued request to run a build according to a specified build definition. Build definitions are submitted to a build queue. A Jazz™ Build Engine receives and processes the build request if its corresponding build engine in Rational® Team Concert™ is listed as a supporting build engine for the build definition.

Build result

A build result represents the output of a build.

Build System Toolkit

Each build computer contains an installation of the Rational® Team Concert™ Build System Toolkit.

The Build System Toolkit consists of the following major components:

Jazz™ Build Engine

The Jazz™ Build Engine is a command-line tool that polls for and processes build requests from Rational® Team Concert™. When the Jazz™ Build Engine is started, it must identify a corresponding build engine in Rational® Team Concert™. TheJazz™ Build Engine can then accept any build request whose build definition is supported by the build engine. The Jazz™ Build Engine runs the Ant script and targets that are specified in the build definition. Each build is represented in Rational® Team Concert™ by a build result.

Build toolkit

The build toolkit is a collection of Ant tasks. Ant scripts can use these tasks to send information (such as build progress, results, links, artifacts) to Rational® Team Concert™ to include in the build result.

Build agent

The build agent is a lightweight process that handles agent-based builds that support z/OS® or IBM® i build scenarios. Jazz™ Build Engine

Note: The Jazz™ Build Engine is a component of the Build System Toolkit; it refers to the process that runs on a build computer and runs Ant scripts.

Application environments

An *environment* is a user-defined collection of resources that hosts an application.

An environment is the application's mechanism for bringing together components with the agent that deploys them. Environments are typically modeled on some stage of the software project lifecycle, such as development, QA, or production. A resource is a deployment target, such as a database or Java™ Platform, Enterprise Edition container. Usually, resources are found on the same host where the agent that manages them is located. A host can be a physical system, virtual machine, or be cloud-based.

Environments can have different topologies, for example: an environment can consist of a single server; be spread over several servers; or spread over clusters of servers. Environments are assigned to specific applications. Although multi-tenant systems can be the target of multiple applications, most IT organizations use application-specific environments. Additionally, approvals can be assigned to specific environments.

HCL DevOps Deploy (Deploy) maintains an inventory of every artifact that is deployed to each environment and tracks the differences between them.

Environment properties can be created with the environment's **Properties** pane (**Applications > *selected application* > Environments > *selected environment* > Properties**).

A value that is set on a component environment property overrides an environment property that has the same name.

Referenced: `${p:environment/propertyName}`.

Creating environments

Before you can run a deployment, you must define at least one application environment that associates components with an agent on the target host.

Before you begin

Ensure that you have at least one base resource to associate with the new environment. See Mapping resources and components to environments on page 2363.

Some of the configuration options might be restricted based on the role assigned to you. Check with your administrator, if you need a specific option enabled.

About this task

The initial environment is typically uncontrolled and often is used to create snapshots. You create an environment for each location to which you deploy an application.

1. Click **Applications**, click your application's name, and click **Create Environment**.
   Result
   The **Create Environment** dialog box opens.
2. Specify the environment name.
   The name is used as part of the deployment process and typically corresponds to the target environment.
   Example
   If you deploy to an integration environment, you might name the environment `SIT`.
3. **Optional:** Provide the following information:

   a. Specify a description.

   b. Assign teams to the environment and set its security type. Click **Add Team**, select values from the **Team** and **Type** lists, and click **Add**.

   c. Enable the **Show Compliancy on Load** option to load the environment's compliancy automatically on the application environment page.
   If disabled, the **Load Compliancy** option will still be available that allows you to load the environment's compliancy on demand.

   d. If you created the application from an application template, in the **Environment Template** list, select the environment template to use for the environment.
   You can assign any security type to this environment.

   e. To ensure that components cannot be deployed to the environment without first being approved, specify the **Deployment Approvals** information. If specified, Deploy enforces an approval process before the deployment can be deployed to the environment. Initial deployments are typically done in uncontrolled environments. After the deployment works as designed, you can configure an approval process as the application moves along the development pipeline. If you are setting up more than one environment, consider creating an approval process for at least one of them.
   You can apply two types of approvals to an environment, that is, external and manual approvals.
      - Specify a **Resource Type to Run External Approval**. You can select **System Default** for the values specified in the system settings, an **Agent Pool** to specify a group of agents, or an **Agent**.
      - Choose an existing **External Approval Process** to run. See Managing external approval processes on page 2392 for information about how external approval processes are created.
      - You can **Exempt Processes** from approvals even if it is required by an environment.
      - Enable **Require Manual Approvals** to allow approvals from users in a role specified by the approval process created for the environment. See Creating an approval process on page 2388 for information about how manual approval processes are created. To prohibit the user who makes an approval request from approving the same request, select **No Self-Approvals**.

f. To prevent changes to snapshots in this environment after you deploy them, select the **Lock Snapshots** check box.

g. If you selected the **Lock Snapshots** check box, from the **Snapshot Lock Type** list, select the lock type:
   - To lock only component versions, select **Only Component Versions**.
   - To lock snapshot configurations, select **Only Configuration**.
   - To lock both component versions and configurations, select **Component Versions and Configuration**.
   - To use the system default value, select **System Default**. The default value is determined by the **Default Snapshot Lock Type** field on the **System Settings** page. For information about system settings, see System settings on page 2086.

h. Select **Require Snapshot** to require that snapshots are deployed to the environment.
   When this option is selected, only snapshots can be deployed to the environment.

i. Select a color to apply to the environment. Default is Bright Cerulean (Blue).

j. To apply the component version and snapshot cleanup settings that are specified on the **System Settings** page, select the **Use Default Artifact Cleanup Settings** check box.
   If this check box is cleared, you must specify how long to keep snapshots and component versions. For more information about cleaning up snapshots, see Cleaning up snapshots on page 2143.

   To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Use Default Artifact Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

k. To apply deployment history cleanup, select the **Use Deployment History Cleanup** checkbox.
   You can use the default settings that are specified on the **Systems Settings** page by clicking **Use Default Deployment History Retention**, or in **Days to Retain Deployment History** you can specify a number of days to retain deployment history records. For more information about cleaning up deployment history, see Cleaning up deployment history files on page 2133.

l. To apply process step logs cleanup, select the **Use Process Step Log Cleanup** checkbox.
   You can use the default settings that are specified on the **Systems Settings** page by clicking **Use Default Step Log Retention**, or in **Days to Retain Step Logs** you can specify a number of days to retain step logs. For more information about cleaning up process step log, see Cleaning up process step log files on page 2145.

4. Click **Save** or **Next**.
5. Click **Create** to submit the request.

Results

The new environment is displayed in the **Environments** pane.

Note: An environment template cannot be deleted if it is being used by an environment.

Creating environments from WebSphere® Application Server cells

You can import information about a WebSphere® Application Server cell into resources and then use those resources in an environment.

Before you begin

You must have a WebSphere® Application Server cell and an application in the HCL DevOps Deploy (Deploy) server.

1. Import resources from the WebSphere® Application Server cell as described in Importing resources from WebSphere Application Server on page 2325.

   Result

   When you are finished, you have a collection of resources that represent the cell and the nodes, servers, and other artifacts in the cell. These resources are under the agent resource for the WebSphere® Application Server system.
2. In the Deploy server, create an application environment.
3. Add the resources to the environment:
   a. Open the environment and go to the **Resources** tab.
   b. Click **Add Base Resources**.
   c. Select the agent resource and click **OK**.

What to do next

Now you can run application processes on the environment, including processes that contain steps in the WebSphere® Application Server - Deploy plug-in.

Deployment triggers

Starting with v7.1.0, deployment triggers replace the run process after a version is created feature.

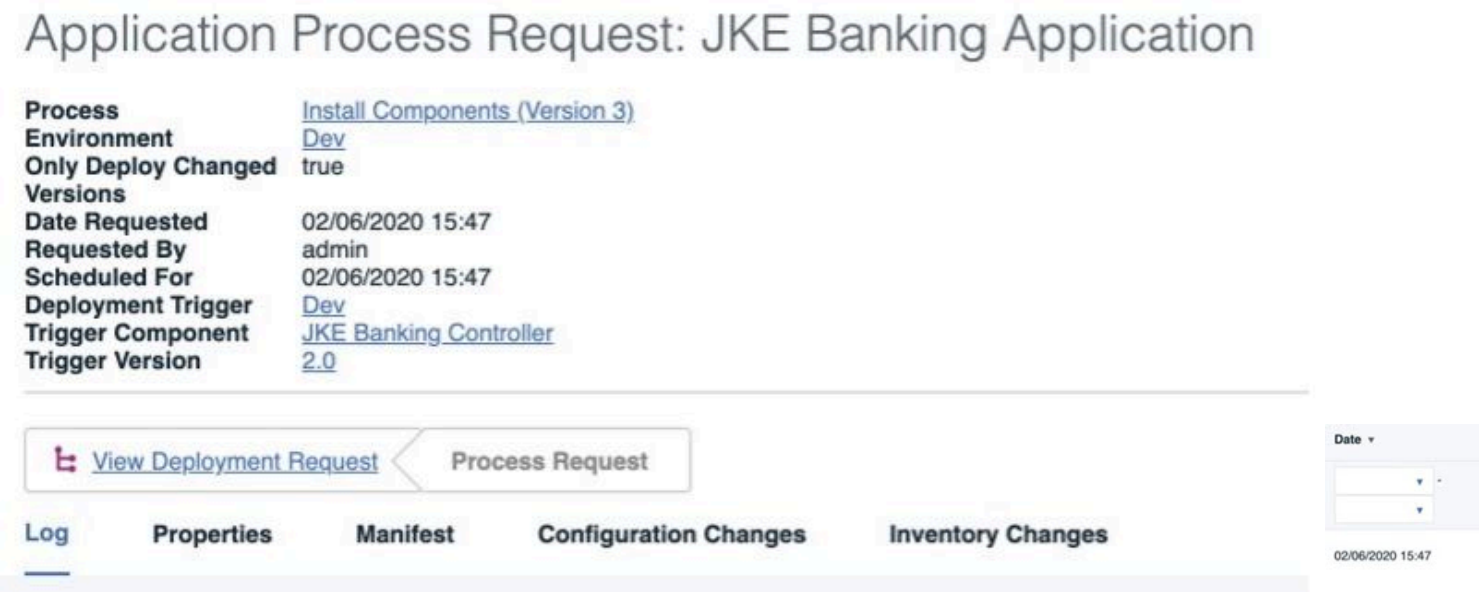
Deployment Triggers are ways for environments to subscribe to certain components. A trigger consists of a specified component, deployment process, and user. Whenever a component finishes importing a version, the server will automatically execute all deployment triggers relevant to that component and will run configured deployment processes against the triggers' environment, using the newly imported version as the deployment payload. With deployment triggers, development environments can be configured to get the latest build artifacts deployed as soon as they are received by the HCL DevOps Deploy (Deploy) server.

When you create a deployment trigger, you have to specify a user who has the run deployment triggers permission, but you do not require that permission yourself. Run deployment triggers is a system level permission that governs whether the a user is allowed to run the process that is the trigger specifies

The user who runs the deployment trigger also must have all permissions that are required to run an application process. Like the run deployment triggers permission, these permissions are also asserted at process run time and not when the trigger is created.

The default audit view is as follows:

- When the deployment trigger is created and who created it
- When the deployment trigger is deleted and who deleted it



Migration

Database upgrades transfer existing deployment triggers to a new table and remove the current constraints, indexes, and columns from the component table.

The specified user for these rows is admin by default.

Importing and exporting

Applications and snapshots that are imported or exported retain information about their deployment triggers.

Importing application snapshots that include component versions do not activate deployment triggers.

Copying an environment retains information about deployment triggers.

Approvals

An application process that requires approvals is still kicked off when a deployment trigger runs the process.

Note: When you update component versions on the command-line interface, a deployment trigger runs when the import is finished.

Before the deployment trigger request submission succeeds submitted, these conditions must be met:

- The component must be mapped to the application when the deployment trigger submits the process request.
- The application process and component must not be deleted.
- The user who runs the deployment must have run deployment triggers permission when deployment trigger runs.

Note: If the user who runs a deployment trigger does not have the required role for a process, then the application process cannot start the deployment by using the deployment trigger.

Creating deployment triggers

With the new deployment trigger feature, you can run an application process automatically on a specified environment after component versions have been added. Deployment Triggers also allow for user specification.

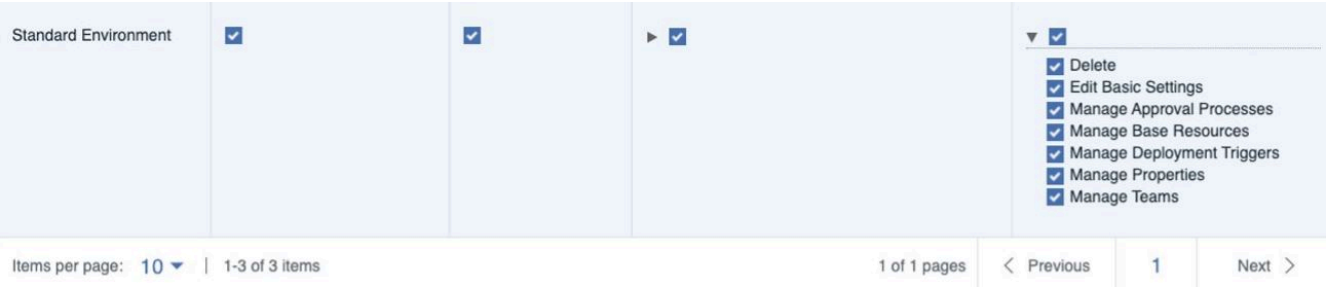
Before you begin

To create a deployment trigger you must have these permissions:

- Edit Basic Settings on environments
- Execute on environments
- Manage Deployment Triggers

To delete a deployment trigger, you must have these permissions:

- Edit Basic Settings on environments
- Manage Deployment Triggers



About this task

You can create deployment triggers for many components on a single environment.

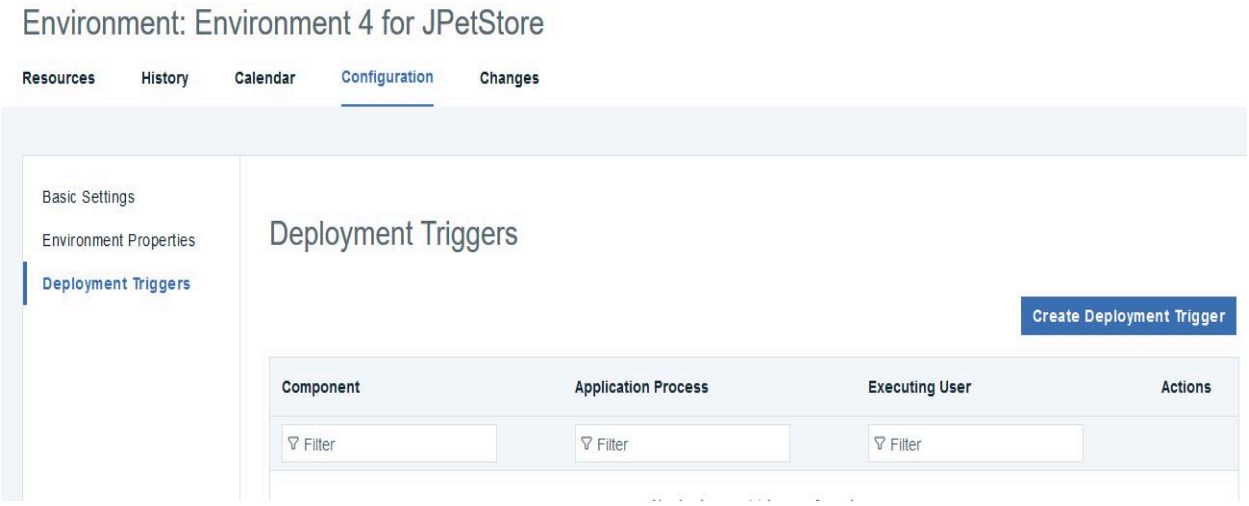
You can run multiple application processes on as many environments as you want.

A deployment trigger adds strict access to running an application process on selected environment through the permission system.

Note: You cannot have duplicate deployment triggers on component, application process, and environment. However, you may create nearly identical deployment triggers when you choose the same component and application process with a different executing user.

Procedure

1. On the Environment Configuration page click on **Deployment Triggers**.



2. Click **Create Deployment Trigger**.
3. In the **Create Deployment Trigger** window, select a component, application process, and executing user.
4. Click **Save** when done.

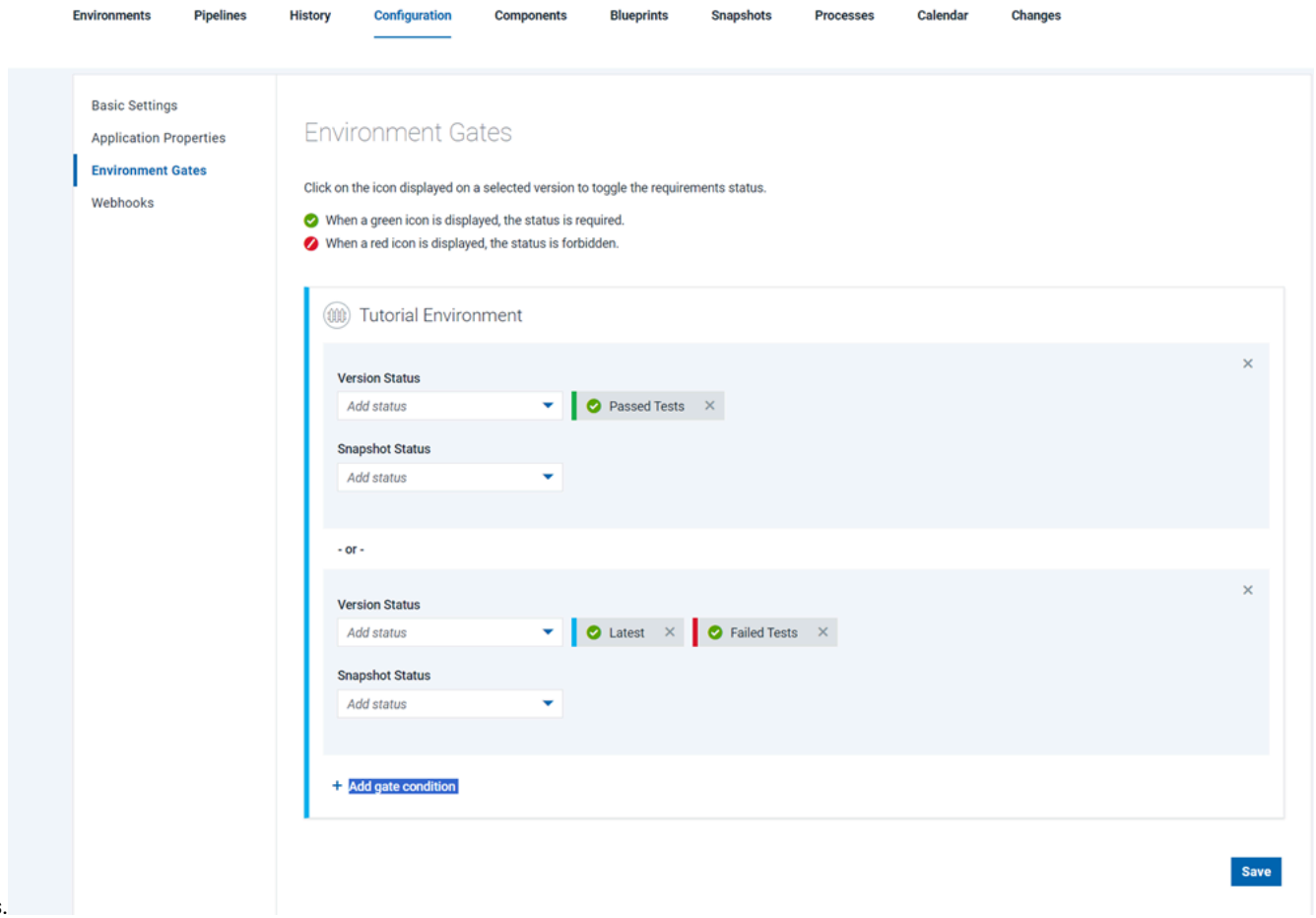
Environment gates

An environment gate is a requirement that must be met before component versions can be deployed to an environment. For example, a component version might need to pass certain tests or receive an approval.

Each environment gate can require a combination of version statuses. These statuses can be combined by using "and", "or" or "not" operators. When the component version has satisfied the statuses and operators, it can be deployed to the environment.

For example, the following figure shows the gate for an environment. The gate has three component version statuses: **Passed Tests** "*or*" both **Latest** "*and*" along with "*Not*" **Failed Tests**. In this case, a component version

cannot be deployed to the environment unless it has the **Passed Tests** status or the **Latest** status and not **Failed**



Tests.

To see the statuses on a component version, click **Components > *selected component*** **> Versions** and look in the **Statuses** field.

Creating environment gates

Create gates that specify the conditions that must be met before a component version or snapshot is deployed to the environment. The conditions are defined on the component version statuses and snapshot statuses.

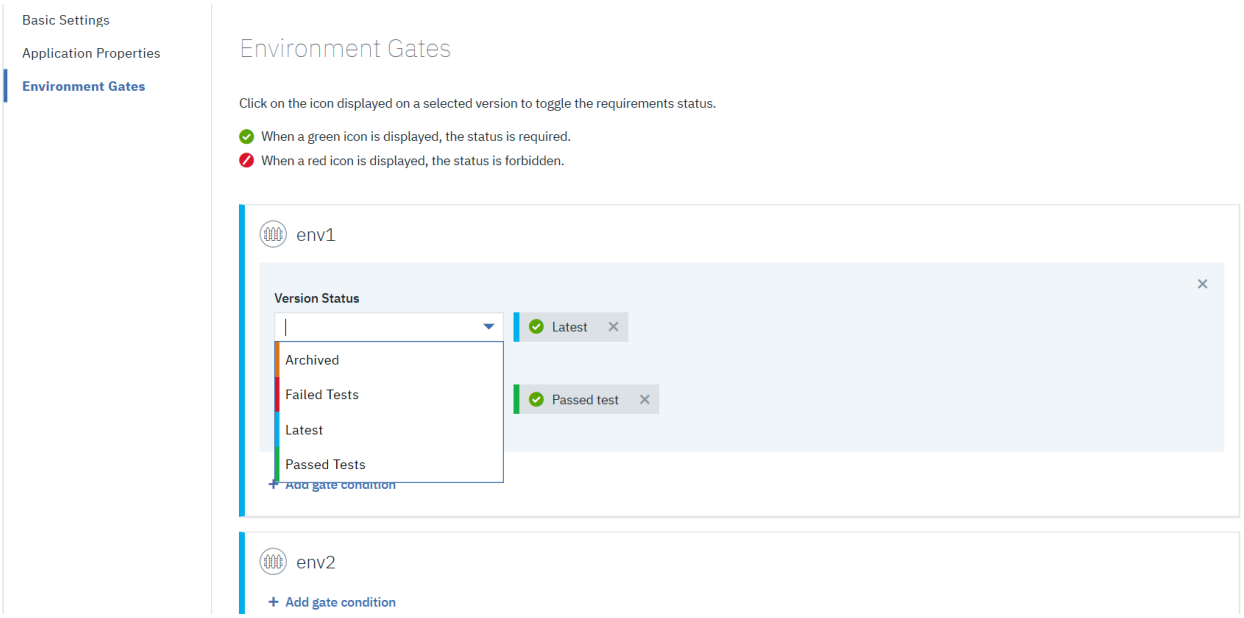
Before you begin

Create one or more component version and/or snapshot statuses. To create statuses, see Creating statuses on page 2243. Note that component versions cannot be deployed to the environments unless they have the specified statuses.

1. Display the **Environment Gates** pane for the target application or application template (**Applications > *selected application*** **> Configuration > Environment Gates** or **Applications > Templates > *selected application template*** **> Configuration > Environment Gates**).
   This pane lists the environments and the gates for each environment.

Note: You cannot modify environment gates if the environment was created by using an application template.

2. Next to an environment, click **Add gate condition**.
3. Select a component version and/or snapshot status by clicking the drop-down arrow from their respective fields.



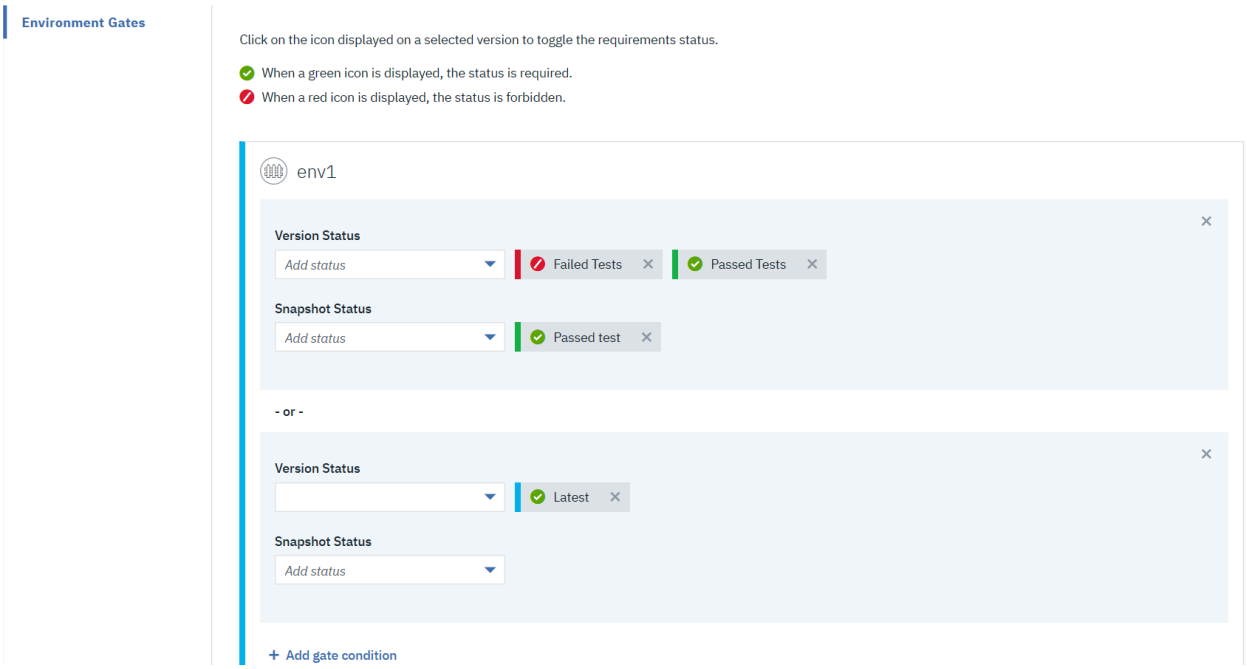
You can add more than one status to a gate condition that must be satisfied for the deployment. For example, if you define a gate with the statuses `Passed unit tests` and `Latest`, only component versions with both of these statuses can be deployed into the environment.

Adding another gate condition, applies the `-or-` operation to the gate. An environment with two or more gate conditions means that the component version can be deployed if it meets any of the conditions. For example, if you define a gate with the statuses `Passed unit tests` and `Latest`, a component version passes the gate if it has either status (or both). A single environment can have any number of gate conditions.

Once you add a status, you can click the green check mark that turns it into a circle-backslash symbol. It gives the status a `Not` operation. A status with circle-backslash symbol indicates that it will exclude the status from deployment. For example. if you define a gate with the condition `(circle-backslash symbol) Failed tests`, the component versions with this status will not be deployed into the environment.

Note: When deploying component versions without choosing a snapshot, the snapshot gates are ignored.

The following example screen capture shows how you can set the gate conditions for an environment.



4. Repeat these steps for other environments.
   Each environment can have different gates.
5. Save your changes.

Results

The environment deployments are executed when all of the specified gate conditions are met.

Mapping resources and components to environments

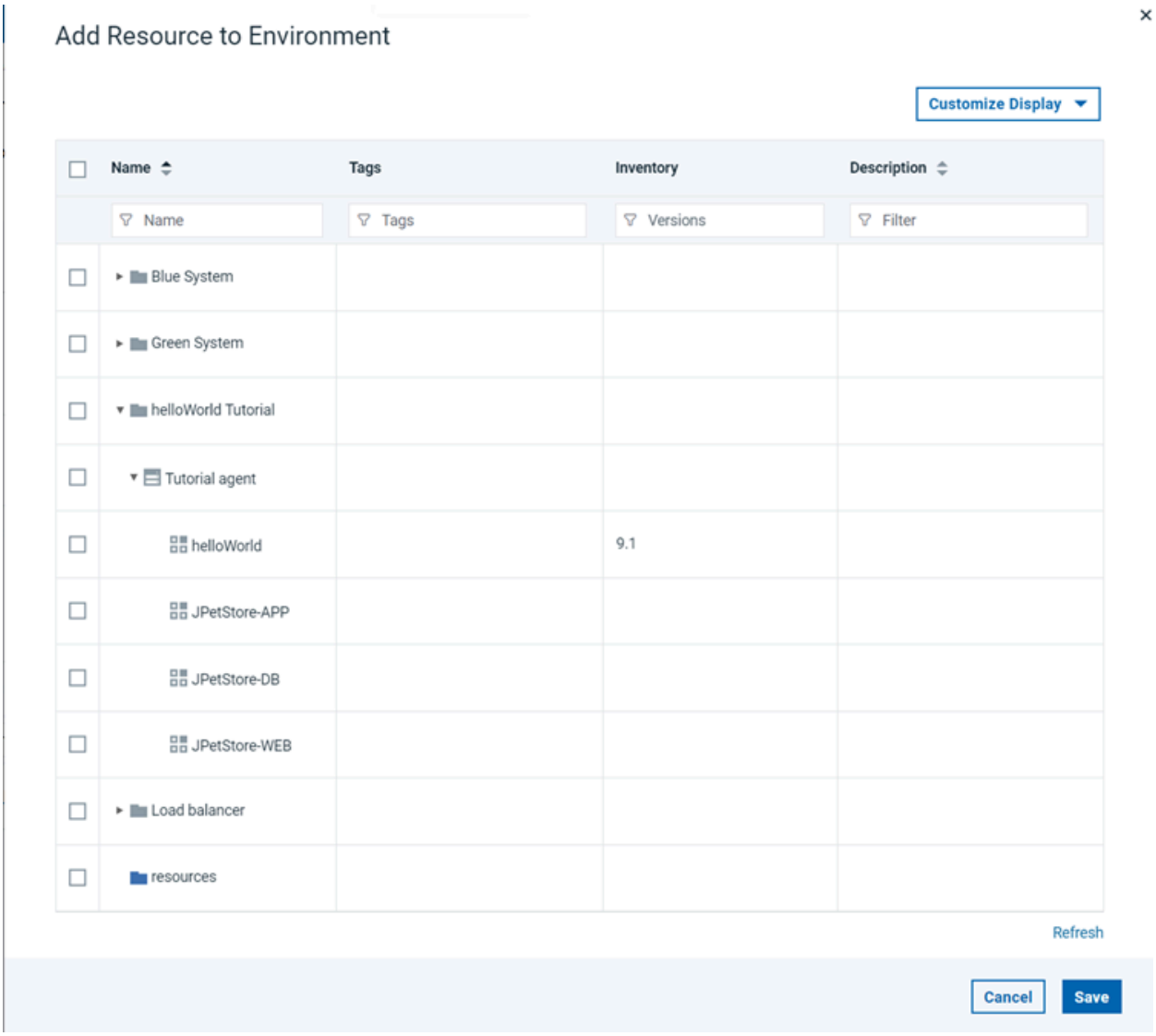
You can specify where components are deployed by adding a resource or resource group to the environment. Then, map the components to those resources.

About this task

Resources and resource groups identify the agents that do the work of deploying components and provide other configuration information. You must identify the resources for the environment and then specify which components run on which resources.

If you are using IBM® WebSphere® Application Server, you can import information from cells as resources; see Importing resources from WebSphere Application Server on page 2325.

1. To open the environment, click **Applications > application > environment**.
2. Click **Add Base Resources** to display the **Add Resource to Environment** window for the selected environment.
   All previously defined resources are available for use, as shown in the following figure:



3. Select the check box that is associated with the resource or resource group to use.
   You can select more than one resource. Selecting a resource that was previously added to the environment has no effect. Adding more than 10 base resources in an environment may result in performance issues.
4. Click **OK**. The selected resource or resource group is added to the environment.
5. To remove a base resource from the environment, in the same row as the resource, click **Actions > Remove From Environment**.
   Removing a base resource removes its child resources as well. Starting from v7.0.5.1, you can delete a resource from the environment. Note that deleting a resource from the environment resource tree deletes the resource from the system. Delete a resource from the system as follows:
   a. Go to **Settings > System Settings**.
   b. Enable **Deleting Resources from Environment Resource Tree**, confirm, and save your changes.
   c. Go back to your environment resource tree, in the same row as the resource, click **Actions > Delete**.
6. Map each component in the application to the resources.

Each component must be mapped to at least one resource. You can map the same component to multiple resources. Or to specify that the component is deployed more than once to the environment, you can map the component multiple times to the same resource. There are several ways to map components to resources:

Choose from:

- To add components one at a time, in the same row as the resource, click **Actions > Add Component**, and then select the component.
- To drag components, click **Show > Components**. Then, in the list of components that are associated with the application, drag one or more components to the resource.
- To add multiple components at once, ensure that each component has the same tag. Then, in the same row as the resource, click **Actions > Add Component Tag**, and specify the tag that identifies the components.

Results

The environment is populated with resources and components.

What to do next

Note: For z/OS deployments, you must use a one-to-one (1:1) mapping between environments and resources. This practice ensures that each resource is assigned exclusively to a single environment within the application context.

After mapping components and resources, create an application process. For information about creating application processes, see Creating application processes on page 2429.

Comparing environments

You can compare two environments to see differences in their inventories or property values.

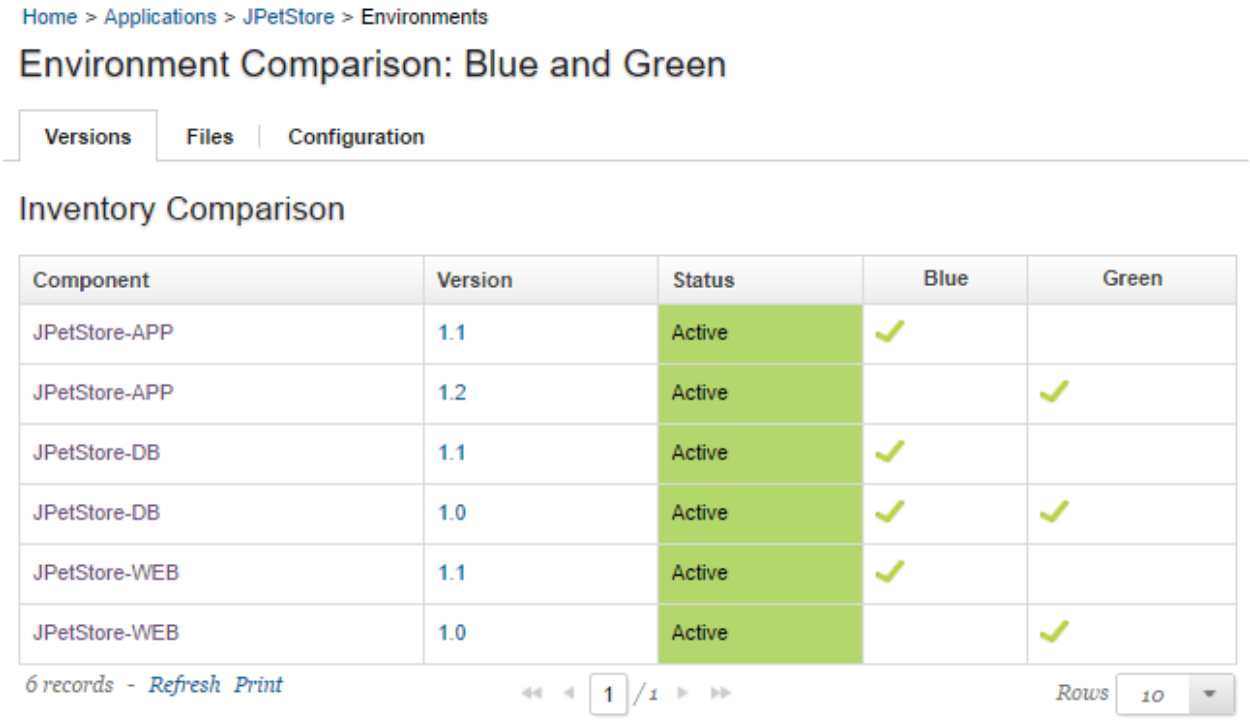
Comparing environment inventories

Comparing two environment inventories shows the differences in component versions between two environments. It also shows the differences in files and properties.

1. From the Deploy dashboard, click **Applications**, and then select an application.
2. Click **More** for one of the environments to compare, and then click **Compare**.
3. From the **Environment** list, select the other environment to compare, and then click **Save**.

   Result

   The **Environment Comparison** page opens. The component versions that are contained in each environment are displayed on the **Versions** tab, as shown in the following figure.



4. To see the differences between component environment properties in the two environments, click the **Configuration** tab.
5. To see the differences between files in the two environments, click the **Files** tab.

Comparing properties

You can run a detailed comparison of properties between environments. You use one environment as a reference point, select one or more other environments, and the server shows the differences in properties on the other environments.

1. From the Deploy dashboard, click **Applications**, and then select an application.
2. Click **Compare Environments**.
3. In the **Select Environments** window, select a reference environment and then select one or more other environments to compare with the reference environment.

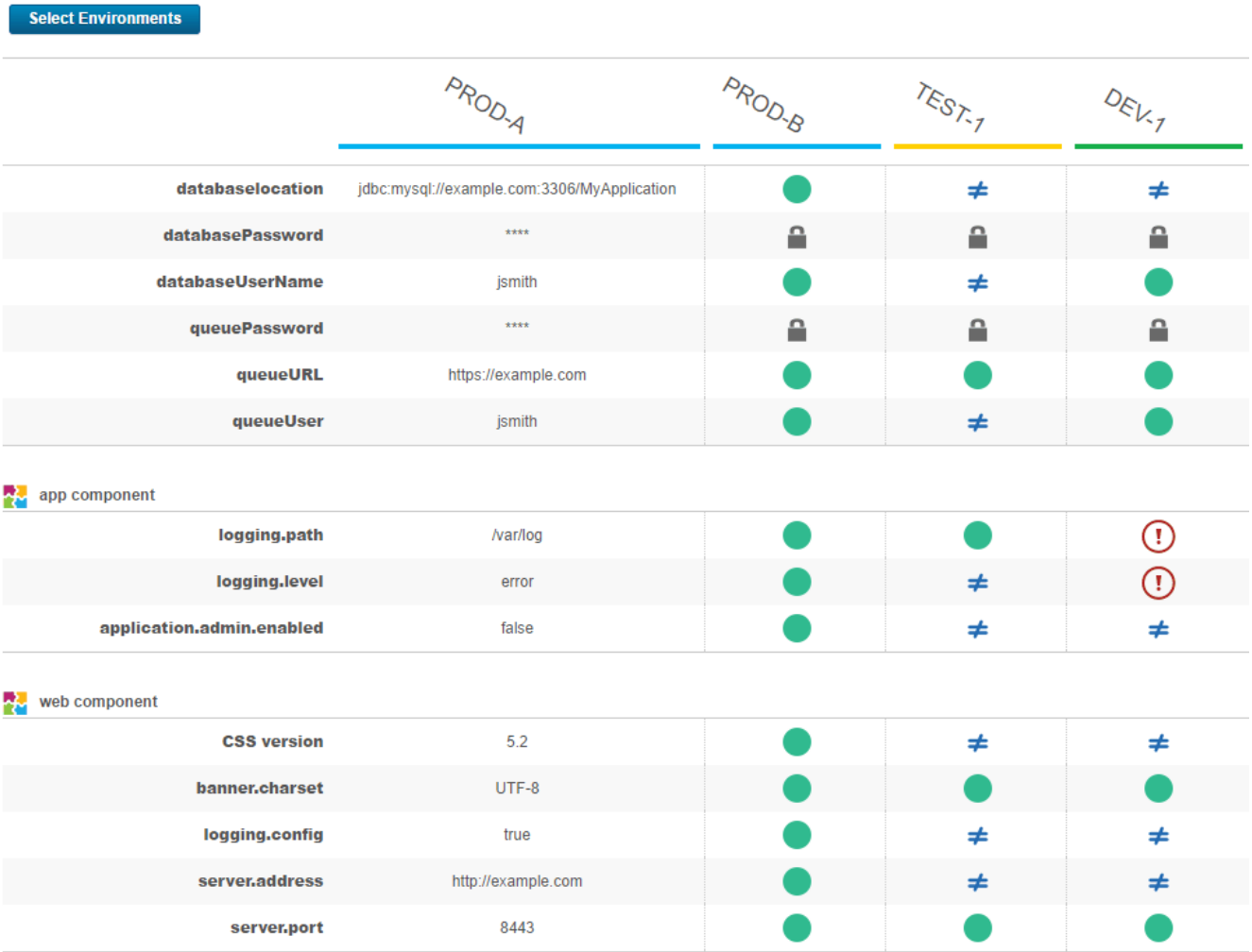
Results

The **Environment Comparison** page shows the differences in properties between the reference environment and the other environments. The reference environment is shown on the left side of the page and the other environments are shown in the columns to the right.

- A green dot indicates that the property has the same value as the reference environment.
- A not equals sign ≠ indicates that the property has a different value than the reference environment.

- An empty circle indicates that the property does not exist on the environment.
- An exclamation point indicates that the property is required, but it does not have a value on that environment.
- A lock indicates that the property is secure. The server does not compare secure properties.

The **Environment Comparison** looks like the following figure:



You can also export the results of the comparison to a CSV file by clicking **Download**. The output file escapes values that start with Excel function characters such as `=`, `+`, `-`, and `@` by prepending an apostrophe. Property values that are longer than 4,000 characters are truncated to 4,000 characters in the output.

Application templates

With an *application template*, you can save and reuse a collection of resources, environments, processes, and tagged components or component templates to create a standardized application.

An application template is a standard framework for users to employ when they create applications. In the application template, you can specify the application properties, environment properties, and environment gates for an application. You can also create a framework that describes attributes of the components, environments, and application processes that an application initially contains. Application templates can contain as many of these elements as required to guide users through the application creation process and to standardize their environments.

Application and environment properties

You specify which application and environment properties are included in applications that are created from the application template. Users must provide values for these properties when they create an application from the application template, and users cannot remove these properties from the application.

Environment gates

You specify which statuses a component version must contain before you can deploy it to an environment. The users cannot modify environment gates in environments that were created by using an application template. To add a version status to a component version, see Adding version statuses on page 2244.

Components

Instead of identifying which components must be used in the application, you can specify which tags that the components that users assign to the application must use.

To control which components users can assign to an application by using the template, you must create a naming scheme for component tags that is relevant to your company. You must also consistently apply the tags to components and component templates. See Adding tags to objects on page 2521.

For example, a simple three-tier web application uses a database, web interface, and application logic. You might tag these components with the following tags: `database`, `web_ui`, and `app_logic`. When you create an application template for this web application, you can specify that the application contain three components, one that uses each of those tags. When users create an application from the template, they can assign to the application one component per tag.

Environments

You can allow users to select environments from the application template. If you do, you must create resource templates that describe the resource tree for each environment. The resource tree can include any agent prototypes, and, if you specify component tags, you can assign the component tags to the agent prototypes. Upon application creation, the components that contain these tags are assigned to the resource tree for the environment. See Creating resource templates on page 2333.

Note: If you create an environment with an environment template and then add component mappings to the resource template, the new component mappings are added to the environment. However, if you

remove component mappings from the resource template, the component mappings are not removed from the environment.

Application process

You can define a simple application process that installs multiple components. Instead of specifying the component and selecting a process, you specify component tags.

To ensure that users can create only applications that adhere to templates, you might create a role for users who create applications from templates. See Creating roles and assigning permissions on page 2188. For more information about role permissions that prevent users from modifying applications to deviate from the application templates, see Application template permissions reference on page 2203.

Note: If users are granted the certain permissions, they can add or remove components, environments, and application templates after they create an application from the template.

Creating application templates

Define application templates so that users can create applications that contain the types of components, the environments, and the resource tree that you specify.

1. Click **Applications > Templates > Create Application Template**.
2. Define the application template:
    a. Enter a name and description, and select the teams that can access the application template.
    b. **Optional:** From the **Notification Scheme** list, select a notification scheme.
    c. **Optional:** To require that a version be provided for each component when a snapshot is created, select **Enforce Complete Snapshots**.
    d. **Optional:** To set the tags that components must use to be assigned to an application during creation, select tags from the **Required Component Tags** list. You specify the minimum, maximum, or exact number of components that contain the tag that the user must select.

        Note:

        By selecting component tags, you limit only the components that a user can assign to an application during its creation. If they have permission, users can add other components to an application after they exit the wizard.

    e. Click **Save**.
3. **Optional:** Define application properties.
   The application properties that you define are included in all applications that use this application template.

a. Click **Configuration**. This **Configuration** tab is associated with the application template that you created, not the main **Configuration** tab.
b. Click **Application Property Definitions**.
c. Click **Add Property**, define the property, and click **Save**.

4. **Optional:** Create an application process.
   The application process that you define is included in all applications that use this application template.

   a. Click **Processes**. This **Processes** tab is associated with the application template that you created, not the main **Processes** tab.

   b. Click **Create Process** to open the **Create an Application Process** dialog box, and create the application process. See Creating application processes on page 2429.

   c. Edit the process in the process editor. See Editing processes on page 2453.

   Note: In the process editor, you can select only a limited number of process steps, such as the **Install Multiple Components** and **Run Operational Process for Multiple Components** steps.

5. **Optional:** Create environment templates.
   The environments that you define can be used by all applications that use this application template. When applications are created from application templates, multiple environments can be created from each environment template. You must assign a resource template to an environment template.
   See Creating environment templates on page 2372.

   Note: To minimize additional steps after you create an application from this template, ensure that the resource template that you select specifies both agent prototypes and the component tags that you selected.

   Note: If you create an environment with an environment template and then add component mappings to the resource template used by the template, the new component mappings are added to the environment. However, if you remove component mappings from the resource template, the component mappings are not removed from the environment.

6. **Optional:** Create environment gates.
   The environments gates that you define must be used by all applications that use this application template and cannot be removed from these applications.
   See Creating environment gates on page 2361.

What to do next

By using team object mapping, grant the team access to application template, agents, component instances, component templates, environment templates, and resource templates. See Mapping teams to objects on page 2215.

Creating applications from application templates

Create an application from a template to apply a set of tagged components, environments, and the specified resource tree to the application.

Before you begin

Confirm that your role has access to an application template and the permissions that are needed to create an application from a template. See Permissions reference for using application templates on page 2203.

1. Click **Applications**, and then click **Create Application**.
2. Complete the information in the **Create Application** dialog box.
   See Creating applications on page 2346.

   Note: You must select the application template from the **Application Template** list.

3. If the application template specifies application properties, you must complete them.
   Application properties are listed at the bottom of the dialog box, after the **Enforce Complete Snapshots** option.

   Note: You cannot delete application properties that you create with the application template. After you create the application, you can view and edit these properties. Open the application, click the **Configuration** tab (not the **Configuration** tab at the top of the page), and then click **Basic Settings**.

4. Click **Next**.
5. Select components from the appropriate lists or create components to satisfy the listed requirements.
   You must select or create the number of components that is specified for each tag type.

   Note: To create components, you must have permission to create components from templates. You can only create components from component templates that contain the tag that is specified in the application template.

6. Click **Next**.
7. **Optional:** Create the environments to use in this application.
   By default, one environment is shown for each environment template in the application template. You can create more environments that use the same template or remove any environments, including those environments provided by default.
   Choose from:
   - To create another environment, click **Create an environment from a template** , and then select the environment from the list. An environment is added to the **Environment list**. The environment name contains the name of the environment template and a unique number.
   - To create a copy of an environment, select the environment and click **Duplicate the selected environment** . An environment is added to the **Environment list**. The environment name contains the name of the

source environment name and "copy." The new environment contains the same values for the description and properties values as the source environment.

- To remove an environment, select the environment from the **Environments** list, and then click **Delete the selected environment** .

Note:

When creating an application from an application template and no environment is selected, a resource named after the application is also created.

Note: You cannot delete an application template if it is in use by an application. You cannot delete environment properties that you create with the application template. After you create the application, you can view and edit these properties. Open the environment, click the **Configuration** tab (not the **Configuration** tab at the top of the page), and then click **Basic Settings**.

8. If an environment template contains environment properties, provide values for them.
   You must provide values for required environment properties. If an environment contains required properties, a blank circle is displayed in the same row as the environment name. When all required environment properties are provided, the circle is shaded.
9. Click **Next**.
10. **Optional:** If you specified environment templates, add agents or agents pools to them.
    Drag an agent/agent pool onto each agent prototype that is shown in the **Environments** pane. If you drag multiple agents onto an agent prototype, a copy of the resource tree for the agent prototype is created for each agent.
11. Click **Save**.

Creating environment templates

When you create an application template, you can define environment templates that model environments for the applications that you create by using the application template.

Before you begin

Ensure that you have a resource template that describes the resource tree for this environment in the application. Include both the component tags that the application templates uses and agent prototypes. See Resource templates on page 2333.

Some of the configuration options might be restricted based on the role assigned to you. Check with your administrator, if you need a specific option enabled.

Create environment templates as follows:

1. Click **Applications > Templates**, select your application template, and then click **Environment Templates > Create Environment Template**.
2. Specify the environment name.

If you create an environment from this template, the environment is assigned this name by default. The name is used as part of the deployment process and typically corresponds to the target environment.

3. Provide the following information:

    a. Specify a description.

    b. **Required:** In the **Resource Template** list, select the resource template to use for the environment.

    c. To set the environment security type, from the **Type** list, select the security type.
    The security type is automatically assigned to environments that you create when you create an application from a template. If you specify an environment template when you add an environment to an existing application, you must manually assign the security type.

    d. To grant teams access to the environment and set the security type for the environment template, click **Add Teams**, select values from the **Team** and **Type** lists, and click **Add**.

    e. To ensure that components cannot be deployed to the environment without first being approved, specify the **Deployment Approvals** information. If specified, Deploy enforces an approval process before the deployment can be deployed to the environment. Initial deployments are typically done in uncontrolled environments. After the deployment works as designed, you can configure an approval process as the application moves along the development pipeline. If you are setting up more than one environment, consider creating an approval process for at least one of them.
    You can apply two types of approvals to an environment, that is, external and manual approvals.
        - Specify a **Resource Type to Run External Approval**. You can select **System Default** for the values specified in the system settings, an **Agent Pool** to specify a group of agents, or an **Agent**.
        - Choose an existing **External Approval Process** to run. See Managing external approval processes on page 2392 for information about how external approval processes are created.
        - You can **Exempt Processes** from approvals even if it is required by an environment.
        - Enable **Require Manual Approvals** to allow approvals from users in a role specified by the approval process created for the environment. See Creating an approval process on page 2388 for information about how manual approval processes are created. To prohibit the user who makes an approval request from approving the same request, select **No Self-Approvals**.

    f. To prevent changes to snapshots in this environment after you deploy them, select the **Lock Snapshots** check box.

    g. If you selected the **Lock Snapshots** check box, from the **Snapshot Lock Type** list, select the lock type:
        - To lock only component versions, select **Only Component Versions**.
        - To lock snapshot configurations, select **Only Configuration**.
        - To lock both component versions and configurations, select **Component Versions and Configuration**.
        - To use the system default value, select **System Default**. The default value is determined by the **Default Snapshot Lock Type** field on the **System Settings** page. For information about system settings, see System settings on page 2086.

    h. Select a color to apply to the environment.

i. To apply the component version and snapshot cleanup settings that are specified on the **System Settings** page, select the **Use Default Artifact Cleanup Settings** check box.
If this check box is cleared, you must specify how long to keep snapshots and component versions. For more information about cleaning up snapshots, see Cleaning up snapshots on page 2143.

To apply the component version cleanup settings that are specified on the **System Settings** page, select the **Use Default Artifact Cleanup Settings** check box. If this check box is cleared, you must specify how long to keep component versions. For more information about cleaning up component versions, see Cleaning up component versions on page 2136.

j. To apply deployment history cleanup, select the **Use Deployment History Cleanup** checkbox.
You can use the default settings that are specified on the **Systems Settings** page by clicking **Use Default Deployment History Retention**, or in **Days to Retain Deployment History** you can specify a number of days to retain deployment history records. For more information about cleaning up deployment history, see Cleaning up deployment history files on page 2133.

k. To apply process step logs cleanup, select the **Use Process Step Log Cleanup** checkbox.
You can use the default settings that are specified on the **Systems Settings** page by clicking **Use Default Step Log Retention**, or in **Days to Retain Step Logs** you can specify a number of days to retain step logs. For more information about cleaning up process step log, see Cleaning up process step log files on page 2145.

4. Click **Save**.

Note:

- If you create an environment with an environment template and then add component mappings to the resource template, the new component mappings are added to the environment. However, if you remove component mappings from the resource template, the component mappings are not removed from the environment.
- Changes in a resource template structure are automatically picked up by the resource sync service but any change in the role properties of a resource template will not be synchronized.

Results

The **Configuration** tab for the environment template is displayed. When you use the application template to create an application, you can create environments by using this environment template.

Exporting application templates

Exporting a template creates a JSON file (file extension `.json`) that contains the template's configuration information, resource templates, environment templates, approval processes, application processes, and generic processes.

About this task

To move application templates between servers, you can export the template and import it on a different server. When you import an application template, all resource templates, environment templates, approval processes, application processes, generic processes, and tags that it uses are also imported.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

Note: Exporting application templates in this way includes only the most recent version of each embedded application process.

For information about JSON, see the JSON website.

1. Go to the **Application Templates** pane (**Applications > Templates**).
2. In the same row as the application template to export, click **Actions > Export**.
   Result
   In the default download location for your browser, a file with the same name as the application template is saved. For example, if the application is named `JPetStore`, the JSON file is named `JPetStore.json`.

What to do next
You can import the application to another server that uses the same version of HCL DevOps Deploy (Deploy). See Importing application templates on page 2375.

Importing application templates

If you exported an application template, you can import the template to a different server. When you import an application template, you can create an entirely new template or upgrade an existing one.

About this task
To move application templates between servers, you export the templates and import them to a different server.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

Note: You must create the JSON file that contains a template on a server that uses the same version of HCL DevOps Deploy (Deploy) as the destination server.

When you import an application template, all resource templates, environment templates, approval processes, application processes, generic processes, and tags that it uses are also imported.

1. Open the **Import Application Template** dialog box by clicking **Applications > Templates > Import Application Template**.
2. Click **Choose File**, and then select the name of the JSON file that contains the template.
3. To upgrade an existing application template, select **Upgrade Template**. To create a template, leave **Upgrade Template** cleared.
   If the name of the application template in the JSON file (not the name of the file itself) matches an existing template name, then the template parameters are upgraded. If items such as processes or properties exist, new versions are created. New items, such as processes, environment templates, resource templates, or properties, are also added. If the name of the application template is not found, the import action has no effect.

Note: The application template name is the first parameter in the JSON file. For example,

```
"name": "JPetStore",
```

4. If the application template's processes contain generic processes, use the **Generic Process Upgrade Type** list to specify how the generic processes that are called in the application processes are upgraded or created.
   This selection affects only generic processes, not the application processes in the application template. The import process has different results depending on whether the generic processes exist.

   Note: The server tracks generic processes by name. If a generic process on the destination server is assigned the same name as a generic process that is imported, then the generic process exists on the server.

   - To import an application template without updating the generic processes that are on your server, select **Use Existing Generic Process**. If the generic process does not exist, a generic process is created. If a generic process with that name exists, another version is not created.
   - To create generic processes, select **Create Generic Process**. A copy of the generic process is always created. If a generic process with that name exists, a unique string is added to the new generic process's name. When run, the application process calls the new generic process.
   - To create a fresh installation and ensure that generic processes are not on the importing server, select **Fail if Generic Process Exists**. If you are creating an application template, it creates both an application template and generic processes unless generic processes exist, in which case the import action fails.

     If you are upgrading an application template, the upgrade fails if the imported application template's generic processes exist.
   - To ensure that generic processes are on the importing server, select **Fail if Generic Process Does Not Exist**. If you are creating an application template, it creates both the application template and generic processes, unless the generic processes do not exist, in which case the import action fails.
   - To upgrade the generic processes, select **Upgrade if Generic Process Exists**. This option creates an application template and versions of the generic processes on the importing server. If the generic processes do not exist, processes are created.
5. In the **Resource Template Upgrade Type** list, specify how to handle resource templates that are being imported, along with the application templates that have the same names as existing resource templates.
   - To import an application template without updating the resource templates that are on your server, select **Use Existing Resource Template**. If the resource templates do not exist, resource templates are created. If a resource template with that name exists, another version is not created.
   - To create resource templates, select **Create Resource Template**. Copies of the resource templates are always created. If a resource template with that name exists, a unique string is added to the new resource template's name. When run, the application process calls the new resource template.
   - To create a fresh installation and ensure that resource templates are not on the importing server, select **Fail If Resource Template Exists**. If you are creating an application template, it creates both the application template and resource templates unless resource templates exist, in which case the import action fails.

If you are upgrading an application template, the upgrade fails if the imported application template's resource templates exist.

- To ensure that resource templates are on the importing server, select **Fail If Resource Template Does Not Exist**. If you are creating an application template, it creates both the application template and resource templates unless the resource templates do not exist, in which case the import action fails.
- To upgrade the resource templates, select **Upgrade Resource Template If Exists**. This option creates an application template and versions of the resource templates on the importing server. If the resource templates do not exist, resource templates are created.

6. Click **Submit**.

Application processes

Application processes, like component processes, are created with the process editor. You use application processes to deploy or to roll back components.

Tip: For full coverage of application processes, see Application processes on page 2428 and its subtopics.

HCL DevOps Deploy (Deploy) provides several common process steps. Otherwise, application processes are assembled from processes that are defined for their associated components.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes, the application determines which ones run and in which order. For instance, an **n**-tiered application might have a web tier, a middleware tier, and a database tier. And, continuing the example, the database tier must be updated before the other two, which are then deployed concurrently. An application can orchestrate the entire process, including putting target servers online and offline for load balancing as required.

Importing specific application processes

Application processes define the steps required to deploy, manage, or rollback specific application. You can import specific application processes to reuse a standardized deployment across multiple applications or environments without recreating a process each time.

Before you begin

You must have ensured that the application process source file to be imported is a JSON or YAML file.

About this task

By importing processes in HCL DevOps Deploy (Deploy), you can incorporate specific deployment logic for applications and reuse them across different environments.

Note: When you export components, applications, processes, or templates, and then import them on a different computer, you must ensure that you install the same versions of the same plug-ins on both computers.

1. Log in to Deploy.
2. Click **Applications**.
   The list of Applications is displayed.
3. Select the application to which you want to import the process.
4. Click **Processes > Import application process**.
   The **Import application process** dialog is displayed.
5. Select **Process Upgrade Type** from the dropdown list.
6. Click **Choose File** to upload the import process source file that you want to import.

   Notes:
   - You must create the JSON file that contains the component definition on a server that uses the same version of Deploy as the destination server.
   - Only JSON and YAML file types are supported.

7. Click **Submit**.

Results

You have imported the specific application process. The imported application process is listed in the **Processes** tab for the application.

What to do next

You can set environment-specific properties by editing the process in the process editor. See Editing processes on page 2453.

Related information

Applications on page 2344

Processes on page 2398

Exporting specific application processes

You can export specific application processes to save and share deployment configurations for reuse across environments.

Before you begin

You must have identified the application processes that are to be exported.

About this task

Exporting application processes creates JSON or YAML file (file extension), that contains processes for the applications. For more information about JSON, see http://www.json.org/. You can export one or more specific application processes as YAML or JSON files.

Note: When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

1. Log in to Deploy.
2. Click **Applications**.
   The **Applications** page is displayed.
3. Click the appilcation from which you want to export specific application process.
4. Click **Processes**.
   The list of associated processes with the selected application are displayed.
5. Export the processes using any of the following methods:.
   - To export processes in bulk:
     a. Select the processes to be exported.
     b. Click **Bulk Actions**.
     c. Select **Export to JSON** or **Export to YAML**.
   - To export a latest promoted process:
     a. Click the menu option of the process to be exported.
     b. Select **Export to JSON** or **Export to YAML**.

Results

You have exported the specified application processes to your local system in the selected file format.

What to do next

After exporting, you can import them into other instances for future use.

Related information

Importing specific application processes on page 2377

Processes on page 2398

Application blueprints

An application blueprint maps application components to a resource template. When you create an environment or provision one from the cloud, you can use the blueprint as a model.

A blueprint begins with a resource template. This resource template specifies a collection of resources, typically including agent prototypes. In the blueprint, you place components on these agent prototypes. The resulting blueprint provides a pattern for an environment, including which components are deployed on which agents.

Creating application blueprints

To create an application blueprint, map the components of the application to a resource template.

Before you begin

Create a resource template as described in Creating resource templates on page 2333.

1. Click **Applications**, click your application, and then click **Blueprints**.
2. Click **Create Blueprint**.
3. Specify a name for the blueprint and select the resource template.
4. Click **Save**.

   Result

   The new blueprint opens, showing the contents of the template, including the agent prototypes.
5. Map the components of the application to the agent prototypes:

   a. Hover the mouse over an agent prototype and then click **Actions > Add Component**.

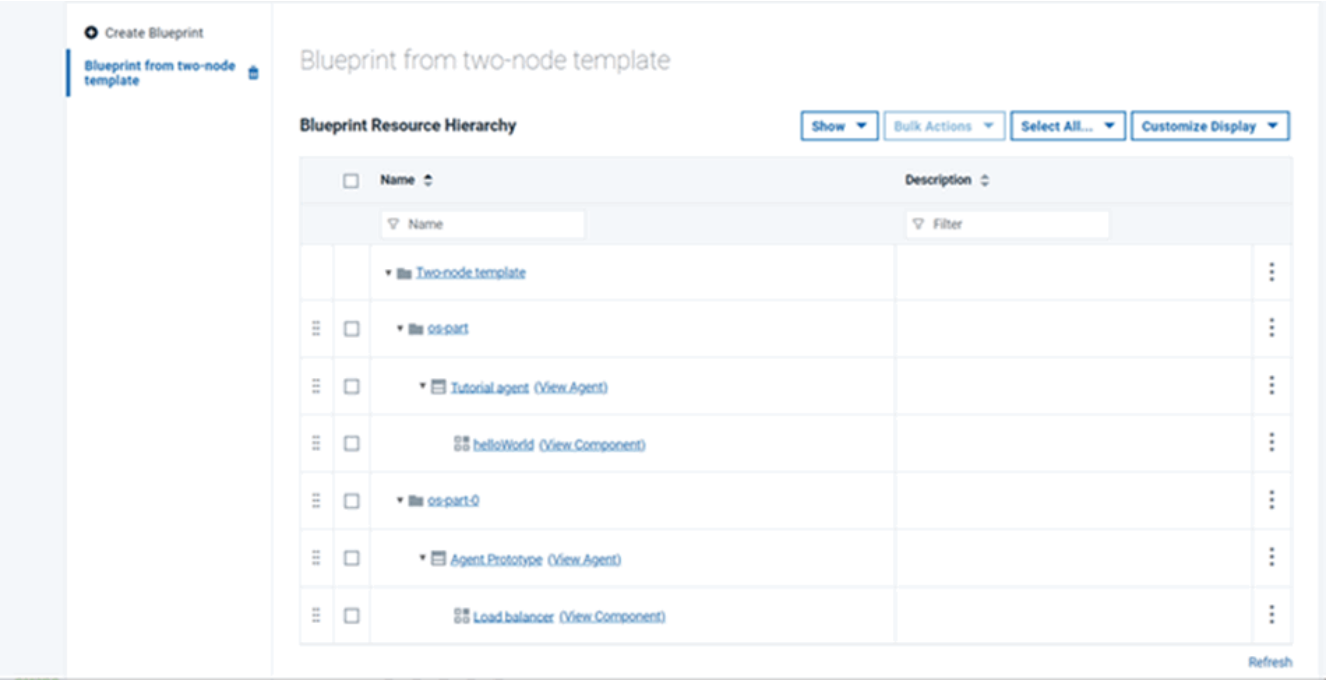
   b. In the window that opens, select a component, specify its information, and click **Save**.

   c. Repeat this process to add the application components to the agent prototypes.
   Depending on how your application is organized, you can add the same component to multiple agent prototypes or multiple components to the same agent prototype.

Note: If you create an environment from this blueprint and then add component mappings to the blueprint, the component mappings are also added to the environment. However, if you remove component mappings from the blueprint, the component mappings are not removed from the environment.

Results

The complete application blueprint shows how the components are mapped to the agent prototypes. The following example shows a simple blueprint. The resource template contains two agent prototypes. In the blueprint, one component is mapped to each agent prototype.



What to do next

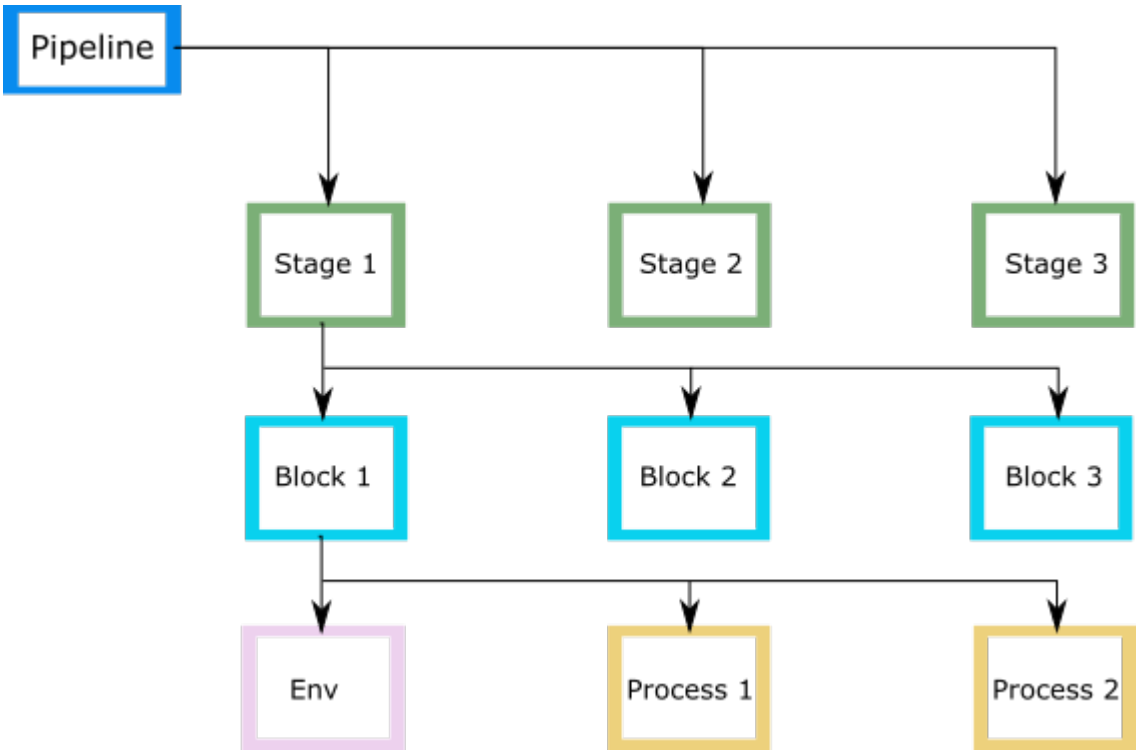
Create environments that are based on this blueprint.

Application pipelines

Allows you to run application processes in multiple environments and promote versions through a set of stages.

Application pipelines includes an ordered list of stages. Each stage in a pipeline is comprised of blocks. A block consists of environments and an ordered list of processes that you want to execute in this block.

The following chart is a graphical representation of an example pipeline definition.



You can create and run multiple pipelines in an application. The pipelines are executed manually.

Managing pipelines

You manage pipelines by defining, editing, deleting, and tracking the execution history of pipelines.

About this task

The structure of a pipeline is defined using JSON format. You can use the following example JSON format to define a pipeline per your requirement:

```
{
    "name": "pipeline-name",
    "stages": [
        {
            "name": "stage-name",
```

```
            "blocks": [
                {
                    "environment": "environment-name",
                    "processes": [
                        "application-process-name-1",
                        "application-process-name-2"
                    ]
                }
            ]
        }
    ]
}
```

Here, `pipeline-name` and `stage-name` are custom names. The `stage-name` can be named to signify the environments you want to deploy. The environment name and application process names must match with the ones existing in the application.

Note: You cannot set properties in the application pipelines. The properties must be preconfigured on the process, environment, and so on.

Manage pipelines from the Applications page as follows:

1. Go to ***application_name*** **> Pipelines > Create Pipeline**.
   The **New Pipeline** window is displayed.
2. Enter a name for the new pipeline and save it.
   The **Code View** tab of the new pipeline is displayed.
3. Define the structure of the pipeline using the JSON format as described above and save it.
   The new pipeline is listed in the **Pipelines** tab. You can manage pipelines by running, editing, or deleting a pipeline. Each pipeline includes its execution history. You can view or cancel a pipeline execution. Note that you can cancel pipelines with Started, Not Started, and Running status only. To run a pipeline, see Executing pipelines on page 2382.

Executing pipelines

After you define a pipeline, you can execute it using the Web UI.

Before you begin

Before executing a pipeline, understand how runtime objects such as stage and blocks act in a pipeline exection:

- Pipeline execution:
  - Stages are executed sequentially.
  - The next stage starts only when the previous stage is completed successfully.
  - A pipeline completes successfully only when all stages are completed successfully.
- Stage execution:
  - Each block in a stage is executed concurrently.
  - A stage completes successfully only when all blocks complete successfully.

- Block execution:
  - Each process is executed sequentially.
  - The next process starts only when the previous process is completed successfully.
  - A block completes successfully only when all processes are completed successfully.

About this task

Execute a pipeline from the Application page as follows:

1. Go to ***application_name*** **> Pipelines > Pipeline Execution** or click on play button next to a pipeline.
2. Select a snapshot or choose a component version that you want to deploy.
3. Click on **Submit**.
   The pipeline executes and shows the runtime objects that further represents the execution status. An execution have statuses such as Not Started, Starting, Running, Canceling, Completed, and Canceled.

Results

The historical record of completed pipelines or any running pipeline execution is listed in the **History** tab. A pipeline execution can be tracked for the details such as Status, Started date and time, Completed date and time, Payload, and Initiated By.

Snapshots

A *snapshot* is a collection of specific versions of components and processes. Typically, a snapshot represents a set of component versions that are known to work together. In most cases, snapshots include all of the components in an application.

A snapshot is typically created when a successful deployment runs in an uncontrolled environment, where there are no approval gates. Snapshots can be created in controlled environments too. When a snapshot is created, a picture of the application's current state is captured. As the application components move from one environment to another, HCL DevOps Deploy (Deploy) ensures that the exact versions and processes that you selected are used in each environment.

Snapshots can help manage complex deployments, which have multiple environments and development teams. For example, after you test and confirm that team A's component works with team B's component, you can create a snapshot. Then, as development progresses, you can create more snapshots to model the effort and drive the entire deployment, coordinating versions, configurations, and processes.

Creating snapshots

To create a snapshot, you specify which version of components and processes to include in the snapshot. Typically, to create a snapshot you copy the components and processes from an environment where you deployed the application without errors.

1. From the dashboard, click **Applications**, and then select an application.
2. On the application page, click **Snapshots**, and then click **Create Snapshot**.
3. Type the name of the snapshot in the **Name** field, and type a description in the **Description** field.

4. Click **Save**.

   Result

   The **Component Versions** page opens for the new snapshot. All the components in the application are listed.
5. Add component versions to the snapshot.

   Choose from:

   - Typically, you create a snapshot by copying versions of components from an environment in which you deployed and tested the application without errors. To copy the component versions in an environment, click **Select For All**, and then click **Copy from Environment**. Select the environment from which to copy component versions and processes. This procedure copies the currently deployed versions of the components from that environment.
   - To create a snapshot that includes the component versions from another snapshot, click **Select For All**, and then click **Copy from Snapshot**.
   - To select component versions individually, for each component click **Add** in the **Versions** column. After you click **Add**, click the **Versions** field to select a version of that component that is available in CodeStation. To select all the latest versions or to select all versions that match a name pattern, click **Select For All** and then select the appropriate option from the list.
6. **Optional:** To refine the versions of the component processes in the snapshot, click **Configuration > Advanced Settings**. By default, the most recently defined version of each component process and property is used. `Latest` is displayed in the **Versions** column, with the actual version in parentheses.
   - To select a specific version of a process or property, click **Set Version** in the **Actions** column.
   - To set all processes and properties to use the latest versions at the time of deployment, click **Reset All to Latest**. In this case, the processes and properties use the most recent versions that are available at the time you run the deployment.
   - To set all processes and properties to use the most recent version that is currently available on the server, click **Set Latest Entries to Current**. In this case, the processes and properties use the most recent versions that are available at the time that you click **Set Latest Entries to Current**. Any changes that are made to processes and properties after this operation are not automatically absorbed into the snapshot.
   - To permanently prevent any changes to the snapshot, click **Lock Configuration**.

     Warning: After you click this button, no changes can be made to the snapshot. This setting cannot be reversed.

7. **Optional:** To lock snapshot component versions, using the **Applications** page, select an application, then click **Snapshot > *snapshot name* > Component Versions**. By default, the lock setting is determined by the option selected in the **Lock Snapshots** field on the **Create Environment** window. To lock previously unlocked component versions, click **Lock Versions**.
8. **Optional:** To lock a snapshot configuration, using the **Applications** page, select an application, then click **Snapshot > *snapshot name* > Configuration > Advanced Settings**. By default, the lock setting is determined by the option selected in the **Lock Snapshots** field on the **Create Environment** window. To lock an unlocked snapshot configuration, click **Lock Configuration**.

Alternately, you can automate snapshot locks within a process by using the **Lock Snapshot** process step. See Lock Snapshot on page 2480.

Note: When you add a new configuration to a snapshot that is locked, that new configuration will not be locked automatically.

Related reference

Lock Snapshot on page 2480

Deploying snapshots

After you create a snapshot, you can deploy the snapshot to an environment.

About this task

Deploying a snapshot deploys the components in the snapshot. The components are deployed in the order in which they were added to the snapshot or to the environment on which the snapshot is based.

1. From the dashboard, click **Applications**, and then select an application.
2. On the application page, click **Snapshots**, and then select a snapshot.
   Result
   The environments that are defined for the application are listed at the bottom of the **Dashboard** page.
3. For the environment where you plan to deploy the snapshot, click the **Request Process** icon.
4. To deploy component versions whether or not they are already deployed, clear the **Only Changed Version** check box. By default, the **Only Changed Version** check box is selected so that only changed versions of components are deployed.
5. Select the deployment process to run from the **Process** list.
6. To run the process in the future, select the **Schedule Deployment** check box.
   Result
   If you select the **Schedule Deployment** check box, more fields become available so that you can specify the date and time to run the process, and configure the process to run on a recurring basis.
7. Click **Submit**.
   Result
   Unless you scheduled the process to run in the future, the application process starts and deploys the snapshot.

Comparing snapshots

You can compare two snapshots to see differences in the files that they contain.

1. From the dashboard, click **Applications**, and then select an application.
2. On the application page, click **Snapshots**.
3. In the **Actions** column, click **Compare** for one of the snapshots to compare.
4. From the **With Snapshot** list, select the other snapshot to compare, and then click **Save**.

Result

The **Snapshot Comparison** page opens. The component versions that are contained in each snapshot are displayed.

5. To see the differences between processes and properties in the two snapshots, click the **Configuration** tab.
6. To see the differences between files in the two snapshots, click the **Compare Files** tab.
7. To see the differences between properties in the two snapshots, click the **Properties** tab.

Previewing snapshots

You can preview an application snapshot before you deploy it to see the upcoming changes in resources and properties.

1. From the dashboard, click **Applications**, and then select an application.
2. On the application page, click **Snapshots**, and then select a snapshot.
3. Under **Environments**, for the environment in which to preview the snapshot, click **Preview**.

   Result

   The **Preview Deployment** window opens.
4. From the **Process** list, select a deployment process.

   Result

   The **Deployment Preview** page opens. The changes to be made to resources and components are displayed.
5. To see the changes to be made to properties, click the **Configuration Changes** tab.

Importing and exporting snapshots

You can import and export snapshots to transfer snapshots and component version artifacts to different servers.

Before you begin

Make sure that the application and the components in the snapshot exist on the target server.

Note: Exporting snapshots in this way includes only the versions of the components that are in the snapshot. However, the exported snapshot always includes the most recent versions of component processes.

About this task

To export a snapshot, go to the **Snapshots** tab for an application and next to the snapshot, click **Export**. The server stores the snapshot, the component versions, and the plug-ins that the components use in an archive file.

To import the snapshot on another server, open the application, go to the **Snapshots** tab, and click **Import Snapshots**.

Note: You must create the archive file that contains the snapshot on a server that uses the same version of HCL DevOps Deploy (Deploy) as the destination server.

Viewing snapshot history

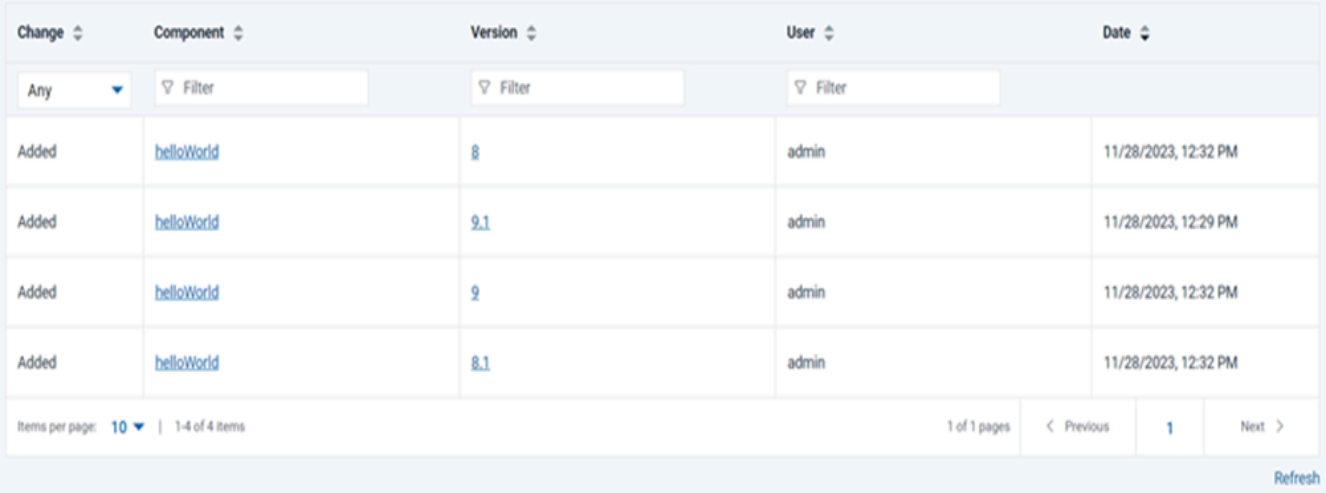
You can track changes to component versions included in a specific snapshot from the **Changes** tab.

Before you begin

Make sure that the application and the components in the snapshot exist on the target server.

About this task

To view a snapshot history, select an application, click **Snapshots > *snapshot name* > Changes**. The **Changes** tab displays the list of changes made to the selected components versions of the snapshot.



You can view the following changes for a snapshot:

- Whether the version is added or removed.
- Impacted component name.
- Impacted component version.
- User responsible for change.
- Time and date of the change.

Manual application tasks

A manual application task interrupts an application process until manual intervention is done. To create a manual task, add a manual step to an application process and select a notification template.

1. Open an application process in the process editor.
2. From the **Utility Steps** folder, add a **Manual Application Task** step to the process.
3. In the **Edit Properties** window, specify the following properties for the step:

Table 731. Manual Application Task properties

Field	Description
Name	The name for the manual task.
Notification Template	Specify a notification template. The individual tasks map to the notification scheme used by the application (see Creating Notifications in a Notification Scheme on page 2081). If you do not want notifications created for the task, select **None**.

Field	Description
Require Comment	Enforces the approver to give a reason for approving or rejecting the task.
Comment Prompt	Type a comment for the approver before they approve or reject the task. You can add properties to the comments. For example, you can use the **snapshot.name** property for approver to identify for which snapshot the approval is required. See Properties on page 2500.
Approver requirements	**Only the deploying user** Restricts approval to the user who started the deployment. **Specific users or group** Allows you to select users or groups that can approve the task.
Add Property	Add one or more properties to the task. You can use these properties later in the process just like output properties from other steps.

4. Click **Save** to save the properties for the step.
5. Include the manual task in the process as with any other step.
6. Save the process.

Results

When you run the process, it pauses at this step, sends a notification, and waits for approval before the process continues. To approve the task and resume the process, click the **Respond** link for the step in the process log, click **Approve**, and then click **Submit**. You can also resume the process from the **Work Items** tab.

Approval Process

You can add approval requirements to deployments. Deployments with approvals cannot be completed until all approvers respond. Approvers can respond to requests on the **Work Items** tab, or by email.

Approval processes are created when you want a manual intervention in executing a particular task or process step. There are two ways in which you apply approval process:

- Generic approval process
- External approval process

Notifications such as an email message can be sent to the approver assigned for each approval process. Configure notification schemes from the Notifications tab of the Settings page.

Creating an approval process

You can add approval requirements to deployments. The approval process identifies the deployment and the role of the approver. You respond to approval requests on the **Work Items** page, or, depending on the configuration, through email.

Before you begin

Before an approval process can be configured, these prerequisites must be met.

- An **ApprovalCompleted** notification template exists for use in the approval process.
- **Require Approvals** is enabled for an environment. In the application environment, select **Configuration** at the task bar and ensure that the **Require Approvals** check box is enabled in the **Basic Settings** window. **Approval Process** will be enabled in the environments task bar.
- The email approval feature is available in notification templates created in V7.1.1 and later. Because templates are not overwritten when you upgrade, you must manually add the email approval feature to your template if you upgrade from a version prior to 7.1.1. To add the feature to your template, paste the following script into your email templates:

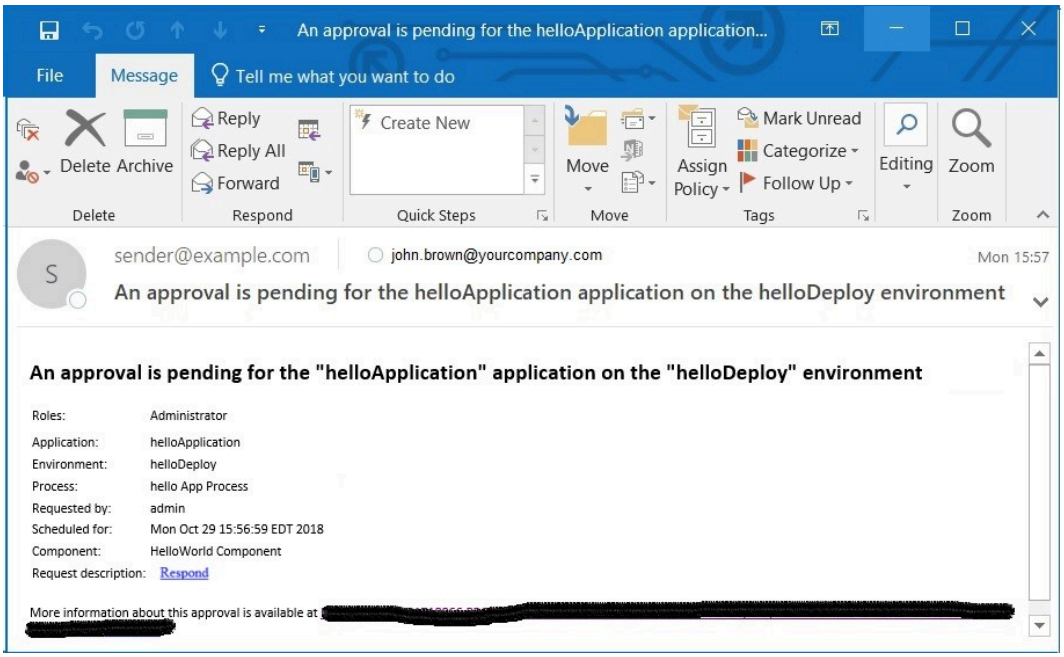
```
#if ($respondUrl)
      <tr>
          <td><a href="$respondUrl">Respond</a></td>
      </tr>
  #end
```

Note: You can add the script to all email templates but currently the `respondUrl` property is only available for TaskCreated and ApprovalCreated notifications.

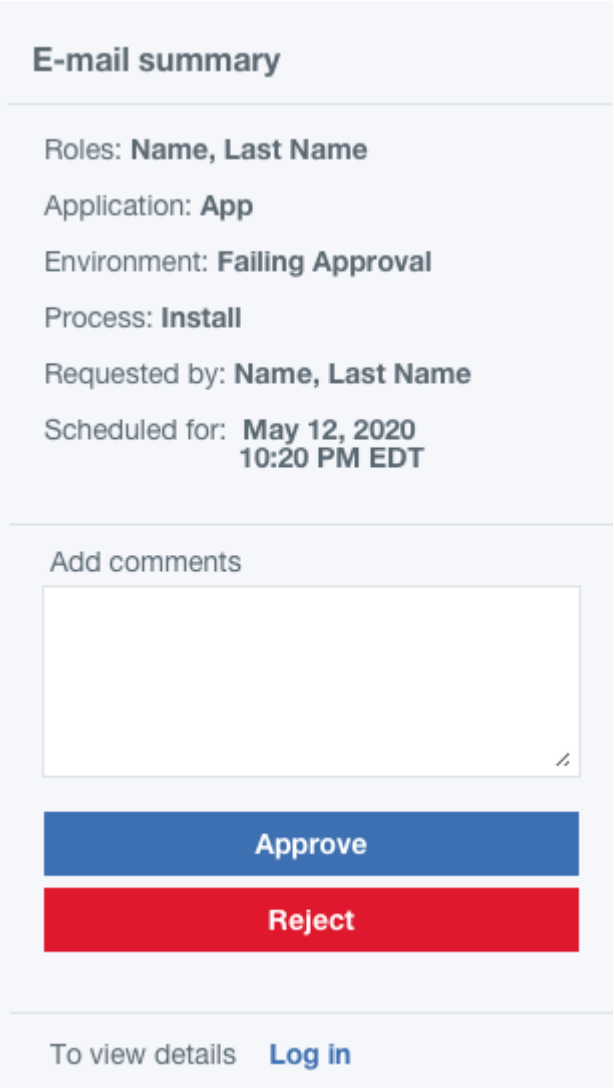
Tip: You can also select the **No Self-Approvals** check box that follows the **Require Approvals** check box. When this check box is selected, the user who sets up of the approval process cannot subsequently approve the process.

1. To create an approval process, display the **Approval Process Design** pane (**Applications > *application name* > Environments > Environment: *environment name* > Approval Process**.
2. Configure the steps that need approval from the process editor. The steps are based on job type and the role of the approver. You have the option of selecting three job types: the application, component, or environment. For help on the process editor, see Processes on page 2398.

   When the process runs, the approver is notified with an e-mail that an approval is required. An example of a default **ApprovalCreated** notification template is shown below.



The e-mail notification includes the **Respond** option that directs the approver to the E-mail summary page. An example of the E-mail summary page is shown below:



This allows the approver to take appropriate action without logging into the Deploy.

Note: The Approve and Reject links expire in 5 days from the date of approval is requested.

Note: If a scheduled deployment that requires approval reaches its start time and the approval is not given, the process does not run and acts as a rejected request. To resubmit a request, you must request a new process. If a process with an approval does not have a scheduled deployment time, the process remains idle until a response is made.

What to do next

- View the status of the request by displaying the **Deployment Detail** pane on the **Reports** tab. If a request is approved, it is displayed as a success. However, if the request was rejected, it shows as failed. If a request is failed, display the **Application Process Request** by clicking **view request**.
- If a comment is made regarding the process, you can view it by clicking the log push button in the actions column on the **Application Process Request**.

Managing external approval processes

If you have a process step that requires validation on an external application before deployment, you can create an approval process and assign it to environments. With this approval process, the deployment runs automatically and only when it is approved.

Before you begin
Ensure that the:

- Notification templates such as Approval Created, Approval Failed, and Approval Completed exist for use in the approval process. For information about how to create notification templates, see Creating Notifications in a Notification Scheme on page 2081.
- **External Approval Settings** in the **System Settings** are configured to use an agent or an agent pool for the external approval process.

1. Navigate to **Processes > External Approval Process**.
2. Click **Create**.
3. Enter a name for the approval process and specify the working directory.
   Optionally, you can provide a process description and add teams to the process.
4. In the process editor, configure the steps that require approval. For help on the process editor, see Processes on page 2398.
   The external approval process definition is complete.
5. Assign the external approval process to an environment.
   a. Click **Applications > *application name* > Environments > Environment: *environment name* > Configuration**.
   b. Enable **Require Approvals**.
   c. Enable **No Self-Approvals** to prevent yourself from approving the process.
   d. Activate **Use Agent Pool for External Approvals** to run the approval process on a group of agents.
      If you turn on this option, specify an agent pool in the **Agent Pool for External Approvals** field.
   e. Select the external approval process to use.
   f. Specify an agent for the approval process in the **Agent for External Approvals** field.

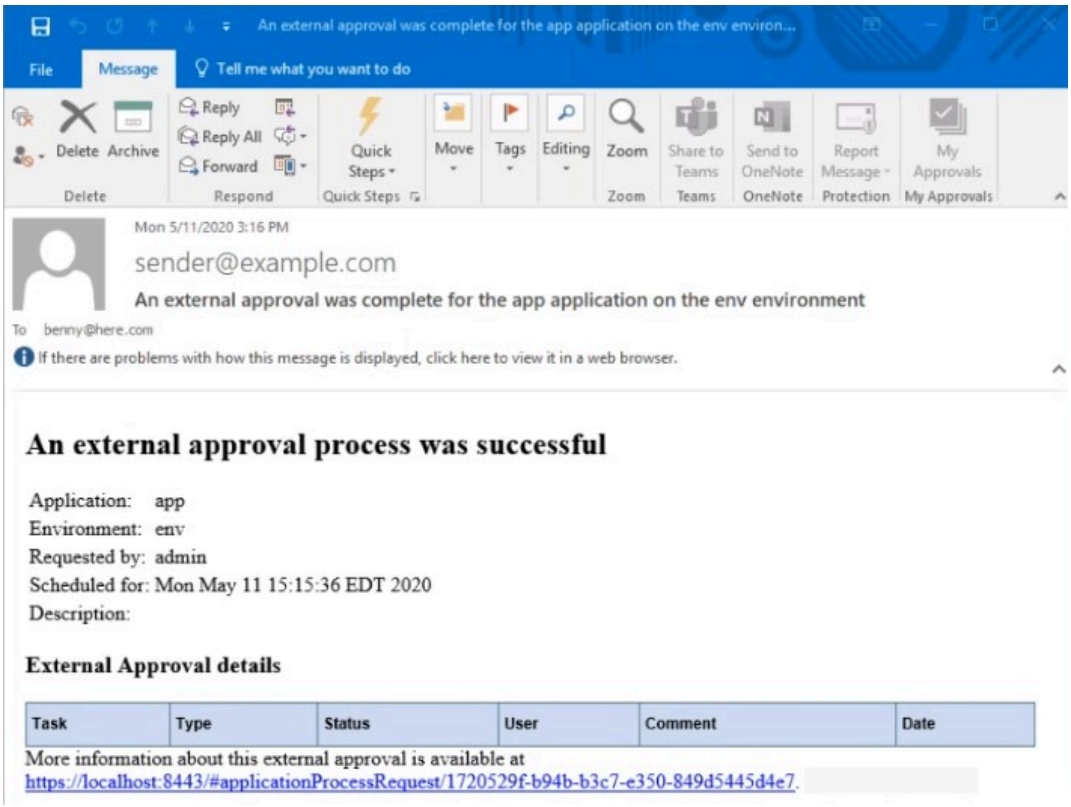
g. Exclude other processes of the environment from the approval process by choosing processes in the **Exempt Processes** list.

h. Save your changes.

The approval process is assigned to the environment.

Result

When an application process runs on the environment, the approver gets an e-mail notification and can approve or reject the request via e-mail. Subsequently, when the request is approved or rejected, the requestor is notified with an email. An example of a default **Approval Completed** notification template follows.



The deployment starts immediately when the request is approved or runs at the scheduled time. If a request is rejected or failed, open the **Application Process Request** by clicking **View Request** and check comments on the approval by clicking the Comment icon in the **Actions** column.

What to do next

You can verify the status of the request by using any of these methods:

- Go to the **Deployment Detail** pane on the **Reports** tab.
- Check the environment history.

An approved request status is displayed with a `Success` label. A rejected request is shown with a `External Approval Failed` label.

Note: While you export or import an application, the approval processes are also included in the export or import.

You can also export and import external approval process individually. While importing, select **Upgrade Process** to import an updated and existing approval process.

Responding to approval requests

If a deployment requires approval, create an approval process that generates an approval request. Approvers respond to the requests on the **Work Items** tab where they approve or deny requests. You can also allow approvers to respond to requests through email.

Until a job is approved, an unscheduled deployment remains idle. If time elapses on a scheduled job that needs approval, the job fails. This control allows the approver to verify the deployment details, and choose the time that it is deployed. Notifications are sent to users who are eligible to complete an approval step if the system is configured with an email server and the user has an email address set.

You can review your approval requests in the **Works Items** tab. This page displays the name of the process, when the request was submitted, who requested the process, and the snapshot or version used. You can also view details of the environment or resource by clicking the link in the **Environment/Resource** column. They can view the details of the target by clicking the link in the target column. Or they can view details on the request by selecting the View Request in the Actions column. The Actions column is also where the approver can respond to the request.

Beginning with version 7.1.1, you can also respond to approval requests through email. An email request provides a link where you can approve or deny the request. The linked-URL does not require a logon to the Web application. If a request requires several approvals, each approver receives a unique link where they can respond.

An open request displays the status as Pending Response. To respond to a request, click actions, and then **Respond**. The approver has the option of leaving a comment. If a request is rejected, the process does not run. If approved, the process begins.

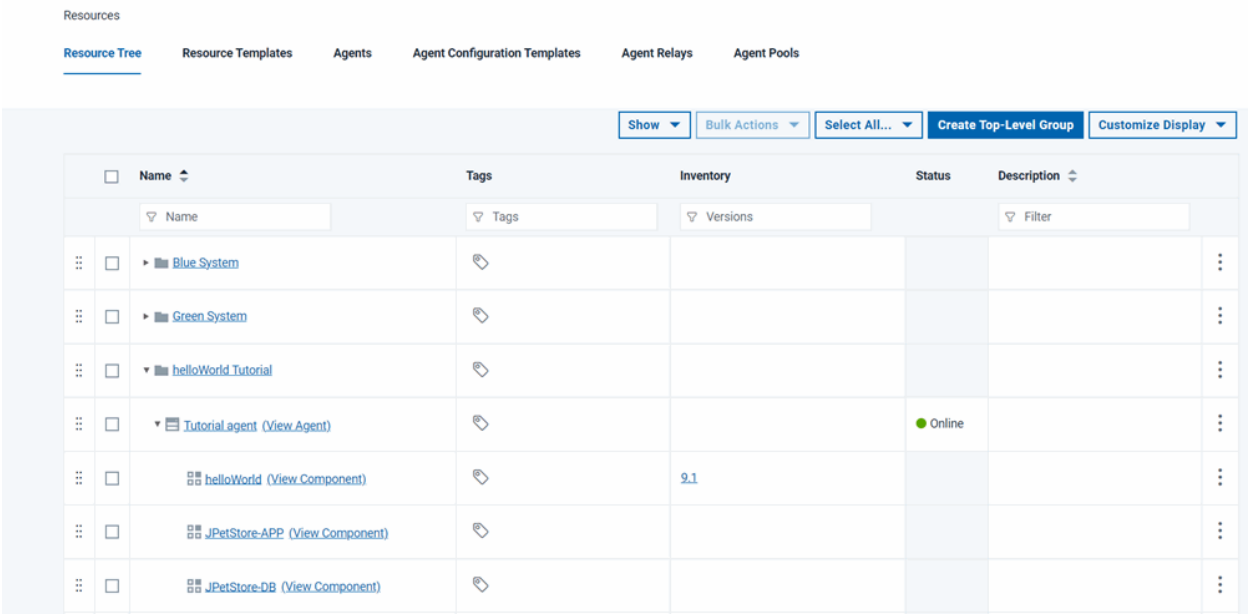
Inventories

You use different *inventories* to track installed components and properties in HCL DevOps Deploy (Deploy) elements.

Inventories provide a complete view of the status and contents of your components. There are different views of the current inventory, depending on your location in Deploy. Inventory information is available on individual components, for every application environment, for each agent resource, and for each component resource.

- The component inventory lists all the resources on which a component is deployed. To see the inventory for a component, click **Components**, click the component, and look on the **History** tab, under **Inventory for Component**.
- The environment inventory lists all the component versions that are deployed to an environment. To see the inventory for an environment, click **Applications**, click the application, and on the **Environments** tab, expand the environment.

- The resource inventory lists all the components that are deployed to an agent resource. To see the inventory for an agent resource, click **Resources**, and expand the resource tree to see which components are installed on which agent resources, as shown in the following figure.



- The resource configuration inventory lists all the properties that are associated with a component resource's elements. To view the configuration inventory for a component resource, click **Resources**, expand the resource tree, select a component, and, from within the component, click the **Inventory** tab. Locate the properties within the **Configuration Inventory**, and click **View** to display the **Property Details** dialog box.

When you create application processes, you have the option of allowing the server to manage the environment inventory and compliance automatically, or to manage the inventory yourself. See Managing environment inventory and compliance on page 2396. Deploy always tracks the component, resource, and resource configuration inventories automatically.

Environment inventory and compliance

The *inventory* for an environment lists the component versions that are intended to be deployed to that environment. *Compliance* shows whether the environment contains those component versions.

The inventory and compliance for an environment are updated each time that you run an application process or deploy a snapshot to the environment. For example, assume that you run an application process that deploys two component versions to an environment. Regardless of whether the application process completes successfully, the environment inventory shows that the two component versions are deployed. The compliance shows how many of the components are installed successfully. In this way, the inventory shows the intended state of a resource or environment, while the compliance shows how closely the resource or environment matches the intended state. By reviewing the difference between the environment inventory and compliance, you can determine whether another deployment is necessary to bring an environment into compliance.

The environment compliance status appears green for compliant environment and red for non-compliant environment. Inventory count is displayed along with the compliance status in the format `number of actual inventory/number of desired inventory`. Here are a few examples of environment inventory and compliance statuses and what they mean:

Status	Description
Inventory: 5/5	The desired inventory is the same as the actual inventory.
Inventory: 2/5 (3 missing)	The desired inventory has more component versions than the actual inventory.
Inventory: 5/5	The desired inventory has the same amount of component versions as the actual inventory, but some versions in the list don't match.
Inventory: 7/5	The desired inventory has less component versions than the actual inventory.
Inventory: 2/2+1	The desired inventory has one fewer version when an attempted uninstall or rollback of a component fails. When a deployment or rollback fails, the incomplete version is removed from the list of desired versions. To troubleshoot component process errors, the inventory display is updated with a "+1" indicator, signifying that a version was removed due to the failure.

The environment inventory and compliance track components but not snapshots or applications. For example, if you successfully deploy a snapshot with two component versions, the inventory is two components and the compliance is two components. If you successfully deploy a different snapshot with versions of two different components, the inventory is four components and the compliance is four components. In this way, the server counts component versions for inventory and compliance, not the versions in a snapshot or applications.

Managing environment inventory and compliance

You can allow the server to manage the environment inventory and compliance automatically, or you can manage the environment inventory and compliance with steps in an application process.

About this task

The inventory for an environment shows the component versions that are intended to be deployed to an environment. The compliance level shows the component versions that are successfully deployed to that environment. For more information about the concepts of inventory and compliance, see Environment inventory and compliance on page 2395.

When you create application processes, you can allow the server to manage the environment inventory and compliance automatically, or you can manage the inventory and compliance yourself. If the inventory management is automatic, the server automatically updates the inventory and compliance for each environment. When you run an application process, the server adds the selected component versions to the environment inventory. Similarly, when component processes run, the server updates the inventory to show which component versions were deployed and which component versions were not.

For example, assume that an application process deploys one component version, but the component deployment process for another component version fails. In this case, the server sets the environment inventory to include the two component versions. The server also sets the compliance to include only the component version that was deployed.

As an alternative, you can manage the environment inventory and compliance manually. In this case, you add steps that make specific changes to the inventory and compliance.

1. In the settings for the application process, set **Inventory Management** to **Advanced**.
2. In the application process, add steps to install, uninstall, or configure component versions as usual.
3. Add the **Component Inventory Update** utility step, and then specify the properties for the step as described in Inventory Update on page 2479.
   Specifically, specify the component to update the inventory for and the changes to make to the inventory and compliance. You can add as many Component Inventory Update steps to the process as you need.
   Example

   For example, to add a component version to the inventory and set the compliance to active, specify the following properties: select the component; from the **Action** list, select **Add Desired Inventory**; and from the **Status** list, select **Active**.

   Note: The Component Inventory Update and Snapshot Inventory Update steps are available only in application processes that have **Inventory Management** set to **Advanced**.

4. **Optional:** If you use snapshots in application processes, add the **Snapshot Inventory Update** utility step. Specify whether to add a snapshot version to or remove a snapshot version from the inventory, as described in Snapshot Inventory Update.

Viewing resource configuration inventories

You can view the configuration inventory for a component, which includes all properties that are associated with the component resource.

About this task

When you deploy a component, you store all the properties that are associated with the component, its application, and its environment on your deployment's resource target. From the resource tree, you can view which properties are active on the component resource.

1. To view the resource tree, complete one of these steps:
   Choose from:
   - To view the entire resource tree, click **Resources**.
   - To view the resource tree for a specific application environment, click **Applications > *application name* > *environment name***.
2. Click the component's name.
3. From within the resource, click the **Inventory** tab.
4. In the **Configuration Inventory** pane, locate the property to examine.
   You might have to expand several levels to locate the properties.
   Example
   To view the properties that are associated with the component, expand both **Components** and the component itself.
5. In the same row as the properties, click **View**.

Result

The **Property Details** dialog box is shown.

6. Review the contents of the **Property Details** dialog box.
   The dialog box contains the following information:
   - The versions of properties that are stored on the resource target
   - The name and value of each stored property
7. Click **Cancel** to return to the resource tree.

What to do next

If the resource target does not contain the correct version of the properties, you can configure a component process to update the properties. See Component process types on page 2423.

Processes

Processes are automated tasks that run on agents.

There are three types of processes:

- Generic processes run outside the context of components or applications. Generic processes are shown on the top-level **Processes** tab. You can use generic processes in these instances, for example:
  - In conjunction with a number of different component processes that do not fit well into a component template
  - When you want to run processes that directly affect HCL DevOps Deploy (Deploy) agents that are unrelated to component, such as changing the agent configuration
- Application processes run within the context of applications. In many cases, application processes call component processes. For example, an application process can call the component processes that deploy their associated components.
- Component processes run tasks on a single component, such as deploying it, uninstalling it, or running configuration tasks on it.

Processes are lists of steps and connections between those steps. Each step is an individual command that runs on a target computer. Steps can manipulate files, run system commands, download files, and run programs. Plug-ins contribute most of these steps; plug-ins are available that work with many different types of software.

You use the process editor to work with processes. See Editing processes on page 2453.

Each time you request a process, the server keeps a log of information about the process and the individual logs from each step. To download these logs, open the process request and click **Download All Logs**.

Process step types and logic

When you design processes, you use different kinds of process steps and joining techniques. You can assign properties to process steps and use them to control step execution.

Conditional processes

Every connection (except connections from the Start step) has a delete tool and conditional flag. The conditional flag sets a condition on a connection.

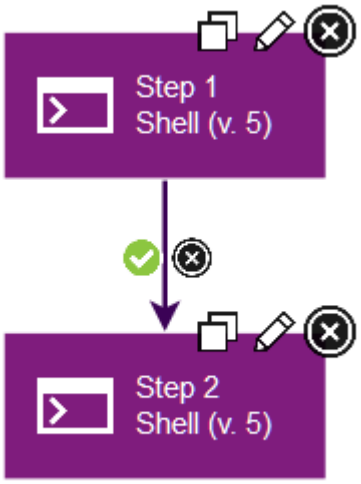
To change a flag value, cycle through possible values by clicking the flag.

The possible conditional flags are as follows:

- `success`: The process is complete.
- `fail`: The process did not finish.
- `both`: The process is complete or the process did not finish. Either status is acceptable, and the next step can start.

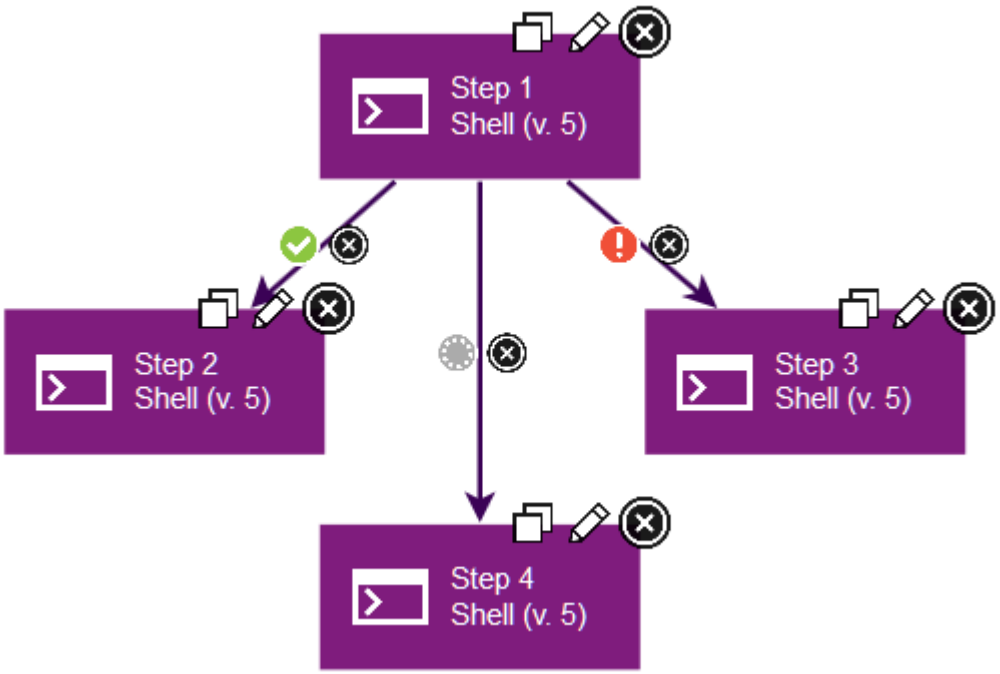
By default, all connections have the flag set to checked (success), meaning the originating step must end processing before the target step starts to process.

The following figure shows two steps that are connected with a `success` conditional flag, which is the default for new connections. In this case, if the first step finishes, the process follows the connection to the second step. If the first step fails, the process ends, and the second step does not run.



A step can have multiple connections with different conditional flags. For example, the following figure shows a step with three connections:

- The connection to step 2 has a `success` flag. Step 2 runs if the first step finishes.
- The connection to step 3 has a `failure` flag. Step 3 runs if the first step fails.
- The connection to step 4 has a `both` flag. Step 4 runs regardless of whether step 1 finishes or fails.



Note: If a step has multiple connections that eventually reach the same target step, determining whether the target runs depends on the value of the intervening flags. If all of the intervening connections have success flags, the target runs only if all the steps are successful.

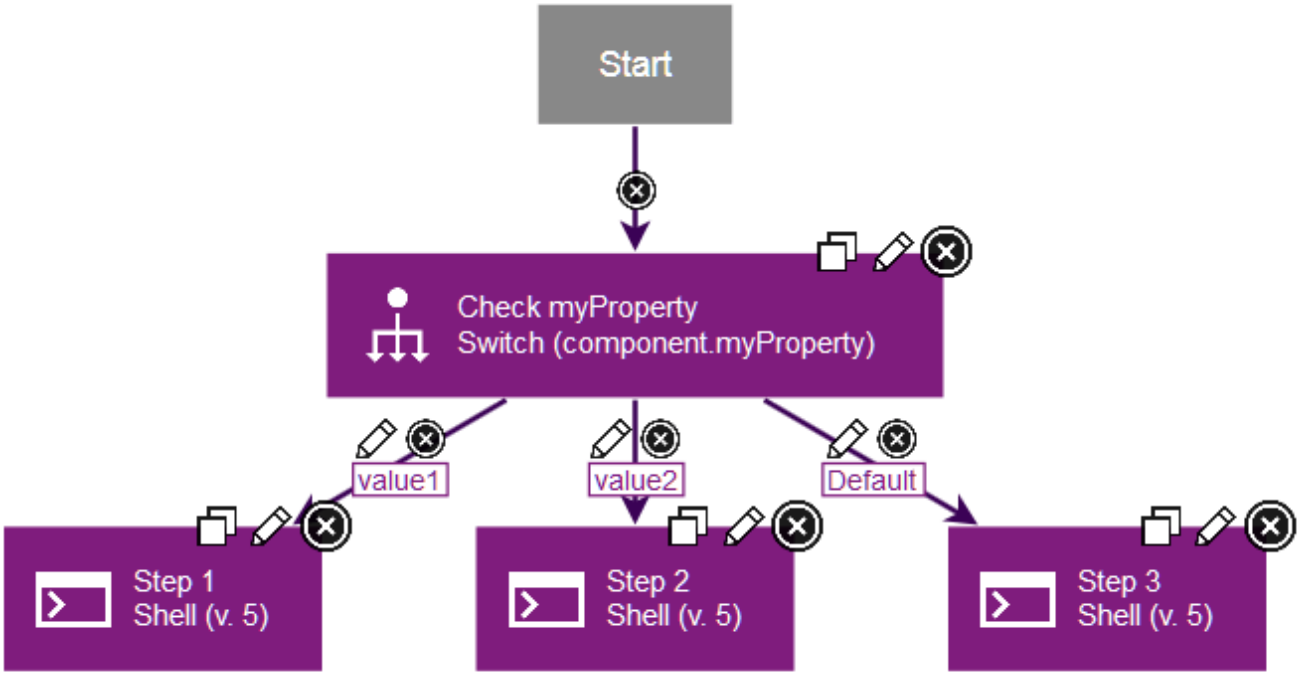
For an entire process to succeed, it must reach a Finish step. If it does not end with Finish, the process fails.

Switch steps

A *switch step* is a utility step that you use to branch process, based on the value of a property.

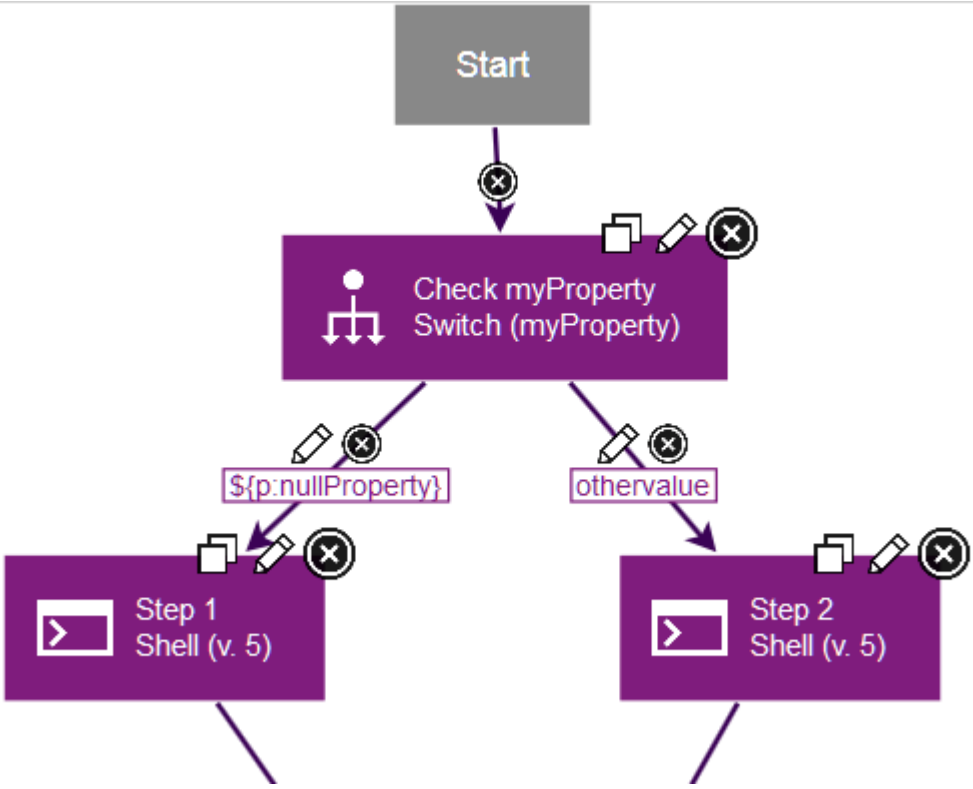
When you add a switch step, you specify a property, either as the property name itself or in the same form as you would use in other steps, such as `${p:propertyName}`. Then, you add two or more outgoing connections from the step, each with a possible value for that property. At run time, the process follows the connection with the matching property value. The process also follows any connections with `fail` or `both` conditional flags.

For example, the following figure shows a switch step for the **myProperty** component property. Two outgoing connections have different values for that property, including `value1` and `value2`. The third connection has the `default` value, which means that the process follows this connection if no other values are matched.



To set the property to use, add the Switch step to the process and click the edit properties icon . In the step properties, specify the property, either by specifying the property name by itself or in the same form as you would use in other steps, such as `${p:propertyName}`. You cannot use a secure property in this step.

To check whether a property is defined, create a second property, and give that property a null value. Then you can compare other properties to that property to see whether they are defined. For example, the following process contains a property that is named **nullProperty**; this property is not defined. In the process, the switch step compares a property that is named **myProperty** to this null property. If the property is not defined, step 1 runs. If the property has the `othervalue` value, step 2 runs.



Pausing a process

You can pause a process to include some updates in an already running process and then resume it. This allows you to apply multiple fixes in between pausing and resuming without requesting a new process.

During the execution of a process with multiple steps, you can press the **Pause** button from the deployment execution log. Pausing a process execution stops the deployment at the next step and results in the following behavior:

- The step which was running, continues to run and retains the Running label.
- The subsequent steps are marked as Not Started.
- The overall process status is marked as Paused at bottom of the page.
- When the current running step terminates processing, as you can see from its output, it still stays in Running state.
- If you click Resume, the currently running step terminates successfully and the next step begins.

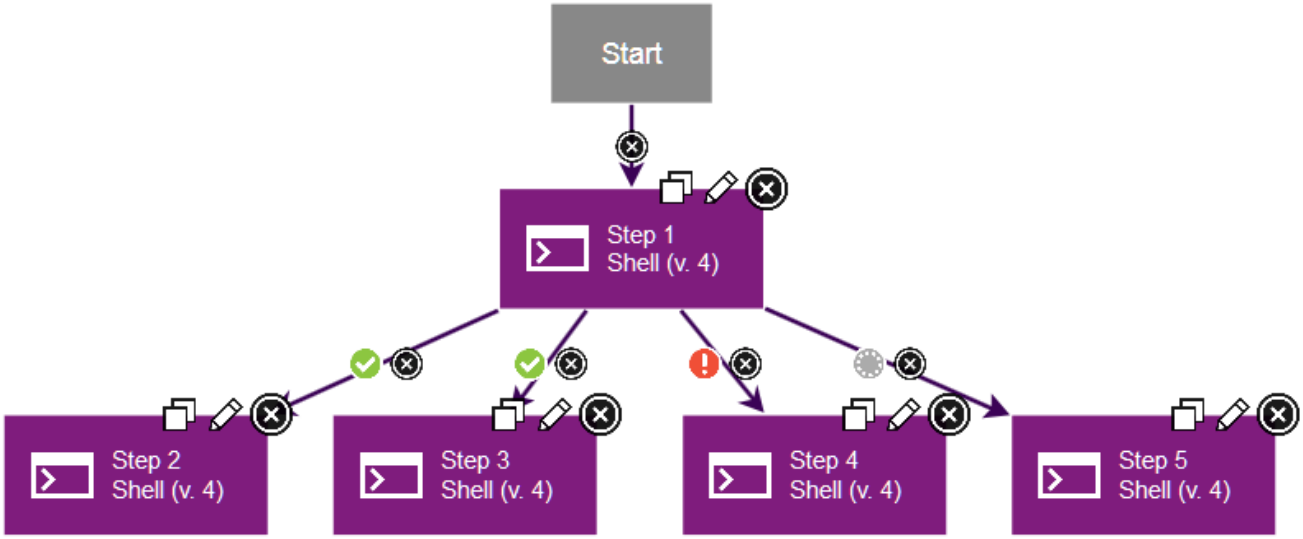
Branching and joining steps

You can branch processes so that multiple steps run at the same time. You can also merge processes to return to running a single step at a time.

Default branching and joining

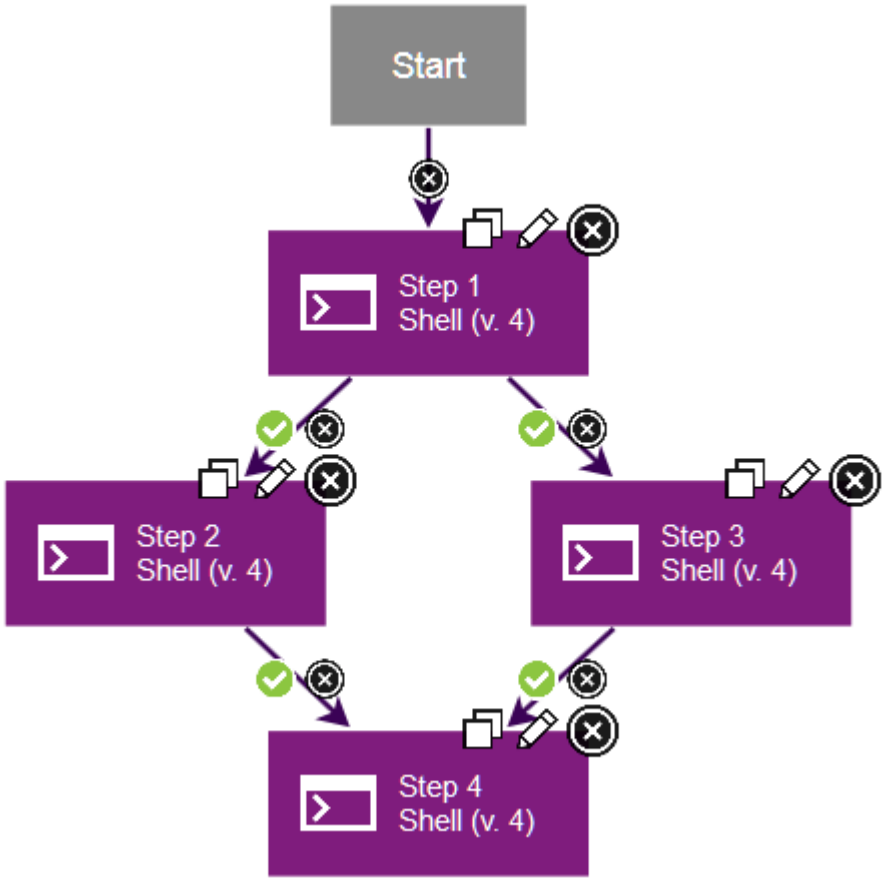
To branch a process and run multiple steps at the same time, add multiple outgoing connections from a step. You can use any combination of `success`, `fail`, and `both` conditional flags, as described in Conditional processes on page 2399. For

example, the following process shows a step that branches into four steps. Two connections have `success` flags. One has a `fail` flag, and one has a `both` flag.

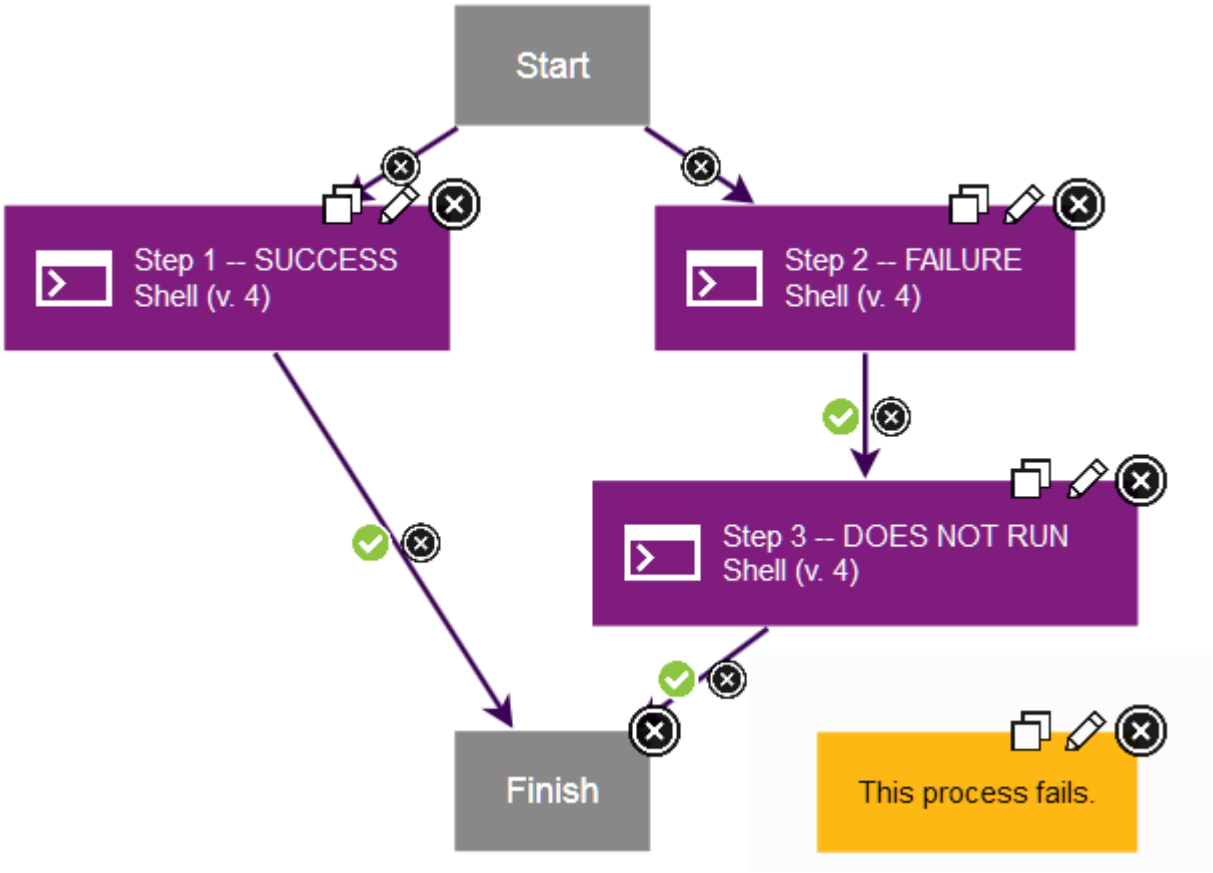


In this case, if the first step finishes, three steps run at the same time: the connections with the `success` flags and the connection with the `both` flag. If the first step fails, two steps run at the same time: the connection with the `fail` flag and the connection with the `both` flag.

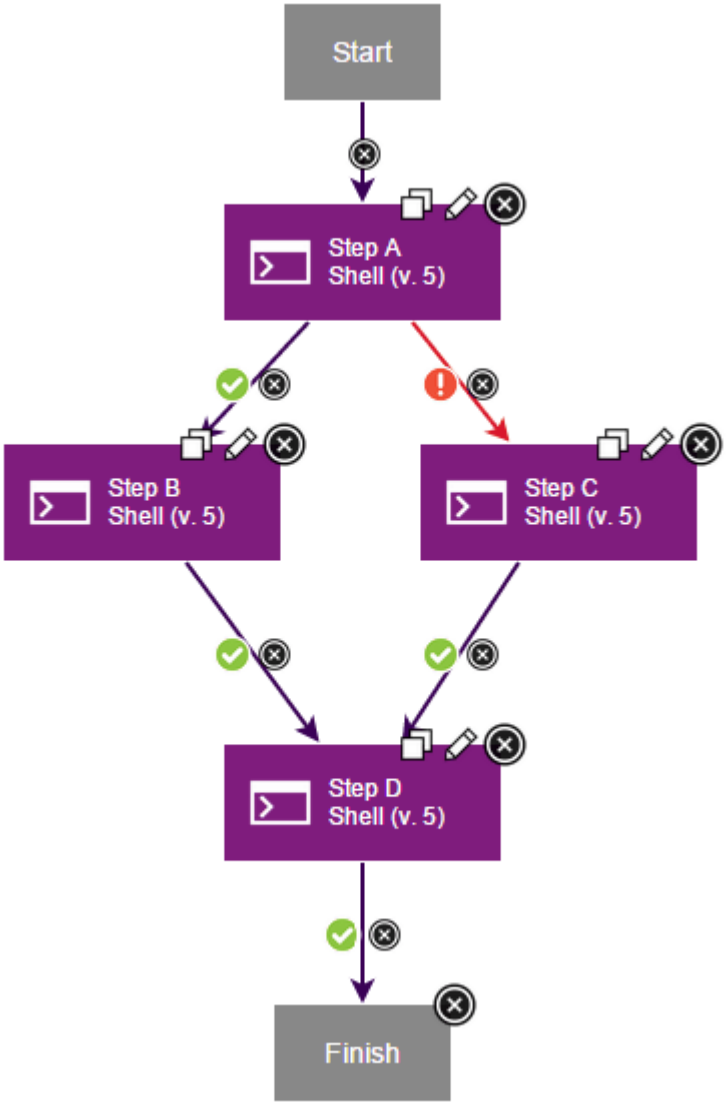
If a process has multiple steps that run at the same time, you can use connections to merge the process into a single step. For example, the following process shows two steps that are running at the same time. Both steps have a connection to a third step, and both of those connections have a `success` conditional flag. In this case, if either step 2 or step 3 fails, step 4 does not run. Because all the parallel processes that branch from step 1 have `success` flags, all the processes that merge at step 4 must succeed for step 4 to run.



How you branch and merge steps can change whether the process finishes or fails. For example, the following process has three steps. Step 1 runs on its own parallel path. In another parallel path, step 2 runs first and then links to step 3. Step 1 and step 3 then connect to the Finish step. If step 2 fails, step 3 does not run. Because step 2 failed, the entire branch is invalid, and the process fails.

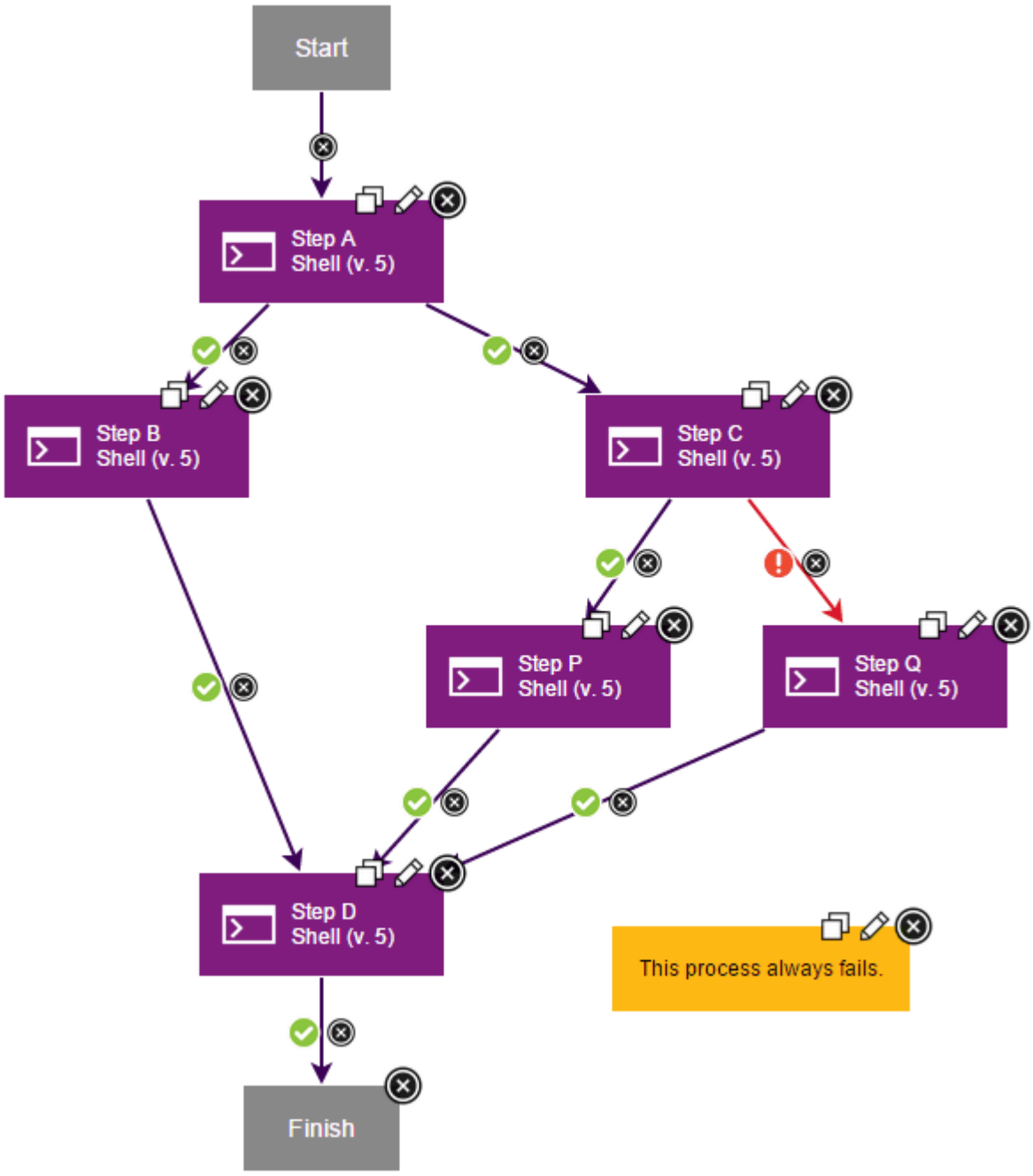


If you assign a `success` flag to one parallel process and a `failure` flag to another parallel process, only one parallel process can run. Because only one parallel process can run, only one of the processes that merge at the merging step must succeed for that step to run. In the following example, if step A succeeds, step B runs. If step A fails, step C runs. Step D runs if either step B or C succeeds.

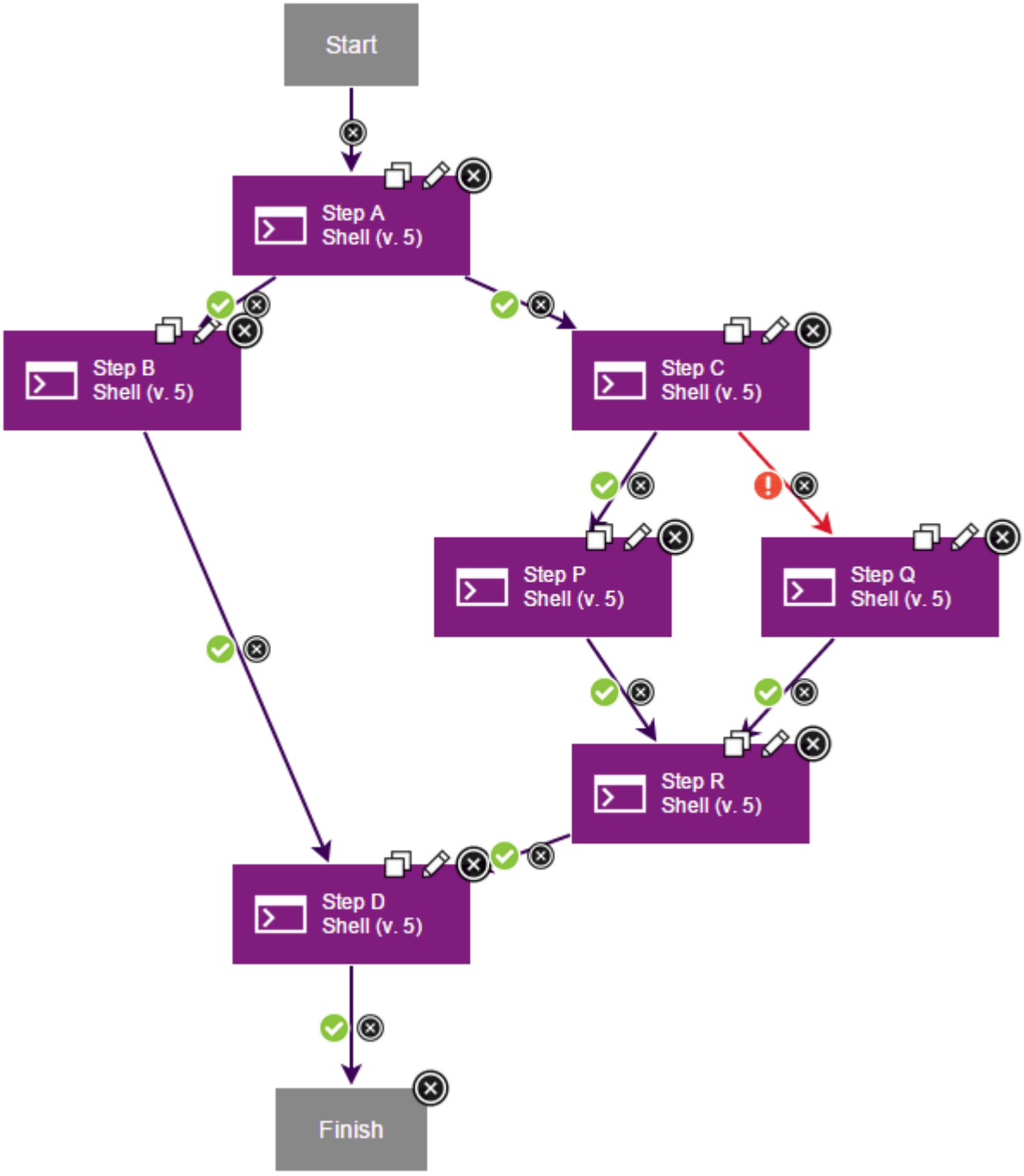


Your process can include nested parallel paths, but you must carefully plan how the parallel paths merge. The rules for the `success` and `failure` flags that you assign at the branching step control how the parallel paths must merge. If you mark all the steps at the branching step with `success` flags, then all the steps that are attached to the merging step must succeed.

The following process contains two branching steps but only one merging step. Because both branches from step A have `success` flags, all the steps that merge at step D must succeed for it to run. Step P runs only if step C succeeds and step Q runs only if step C fails. Because both step P and step Q merge at step D, it is impossible for all the steps that join at step D to succeed. This process always fails.



The following process adds a separate merging step to resolve the parallel paths that branch at step C. As in the previous example, either step P or step Q runs. Unlike the previous example, steps P and Q merge at step R, which follows the rules for its branching step and runs if either step P or Q succeeds. Step D still follows the branching rules for step A and runs only if all the processes that it merges succeed. If both step B and step R succeed, step D can run. By resolving each set of parallel paths separately, your process runs as planned.

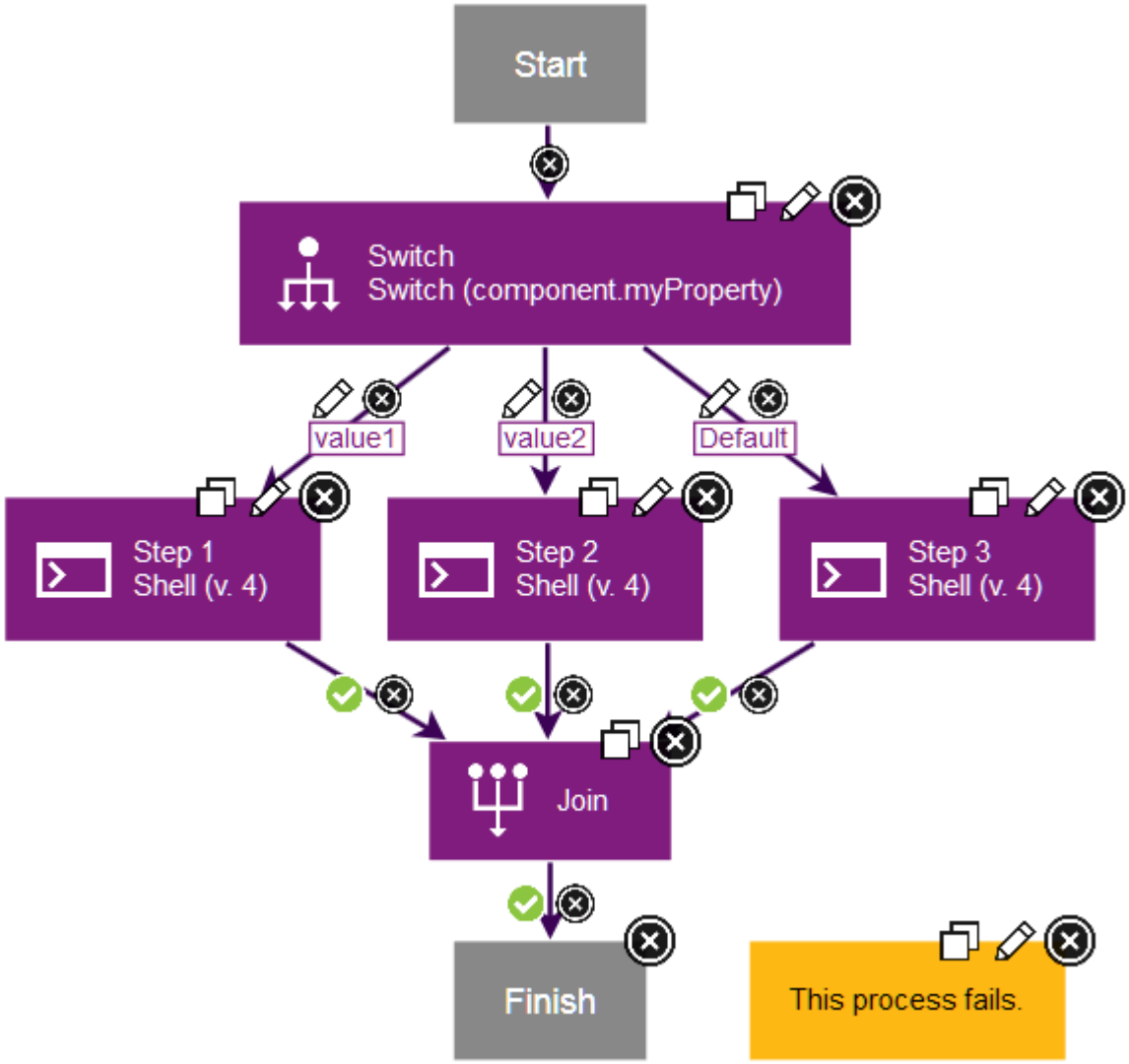


By carefully planning the location of your branching and merging steps, you can ensure that your processes run correctly.

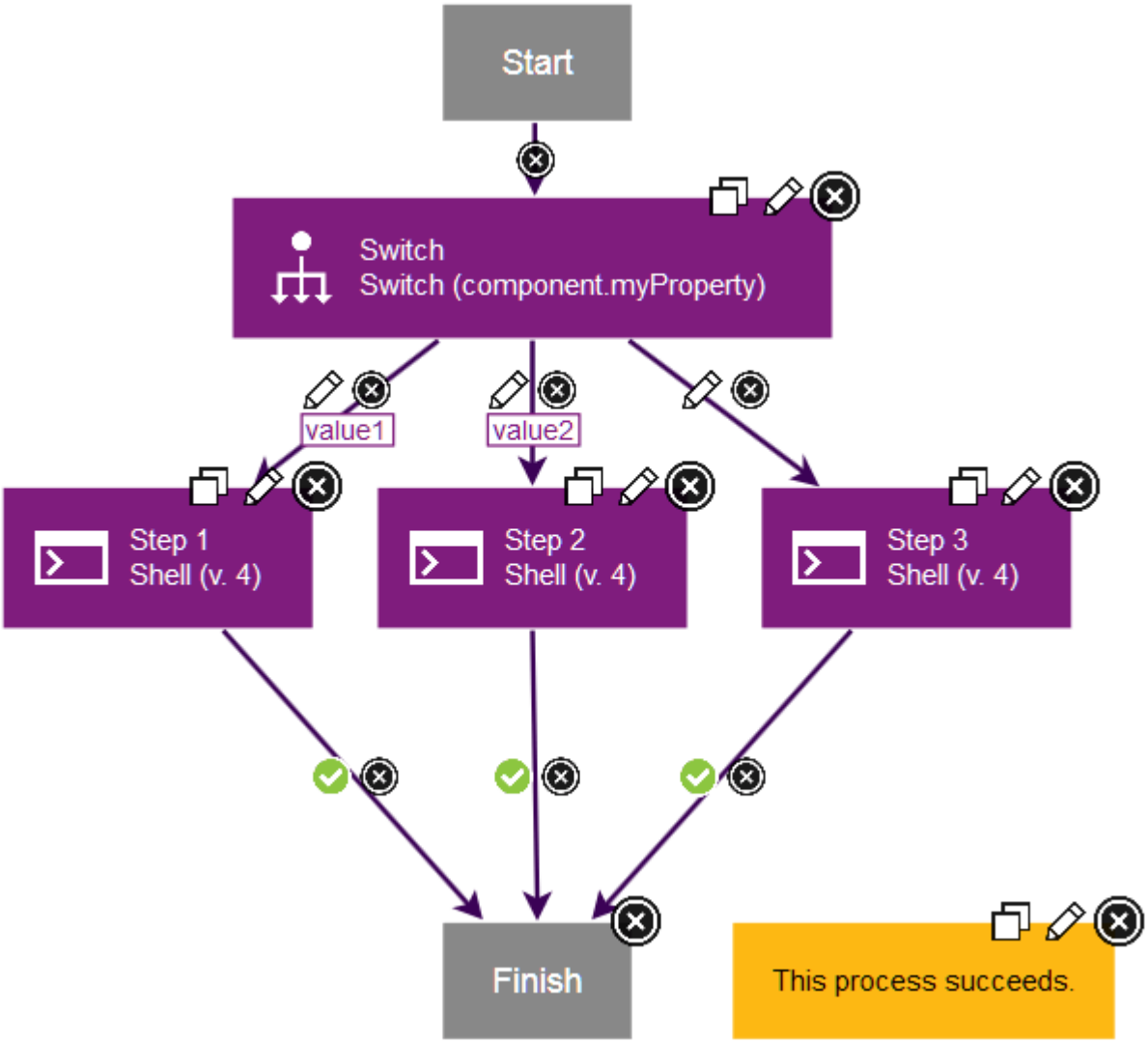
Branching and joining with the Switch and Join step

Use a Join step to explicitly indicate that all connections to the Join step must succeed. The Join step fails if any of the connections to it fail or do not run. For example, the following process uses a Switch step to branch the process. The Switch step has three outgoing connections, each for a different possible value of a property. Thus, only one of these outgoing

connections runs. The three outgoing connections are merged at a Join step. The Join step always fails because only one of the outgoing connections runs. The two other step chains do not run, and therefore the Join step fails.



In these cases, when you do not intend for each step chain to run, do not use a Join step. Instead, merge the step chains by connecting the chains to a single step, as shown in the following process.



Process step properties

All steps have the following properties: `exitCode`, `status`, and `lines of interest` (LOI, which are items that the post-processing script finds in the step's output log).

You can view the properties by using the component's **Log** pane to examine the step's output log (**Components > *selected component*** **> Component request history > View Request > Properties**).

Inventory and versions statuses, which are defined with the **status** property, can be used in application approval gates (see Environment gates on page 2360). The other properties can be used by post-processing scripts, see The post-processing element on page 2576.

Process step preconditions

Most steps in a process can have a precondition. If the **Precondition** field is available, the process supports a precondition. This precondition determines whether the step runs. Like post-processing scripts, preconditions use JavaScript™ 1.7.

Note: The syntax for preconditions is similar to the syntax for post-processing scripts. See Post-processing scripts on page 2415.

Preconditions return Boolean values. If the precondition returns `True`, the step runs as usual.

If the precondition returns anything other than `True`, the process skips that step. In the process request, the step is labeled "Precondition Failed." However, in this case, the process continues to run; the process follows the success links from that step as though the step succeeded.

The precondition has access to the same properties that the step does. Application processes have access to the following properties:

- Application properties
- Application process properties
- Environment properties, including component environment properties
- System properties
- Output properties from previous steps. See Output properties on page 2516.

Property availability depends on your context. See Property availability on page 2519 for information about which properties you can access.

After you run a process, you can see the properties that the process had access to by opening the process request and clicking the **Properties** tab.

To access a property, use the properties.get() command. For example, the following code retrieves the value of an application property:

```
properties.get("application/property1")
```

The following example precondition allows the step to run only if the application property `property1` exists and is not null:

```
properties.get("application/property1") !== undefined &&
properties.get("application/property1") !== null &&
properties.get("application/property1") !== ""
```

Important: Do not use notation such as `${p:component.myProperty}` in the precondition script. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`.

Precondition scripts can access the process properties. For example, if the process has a property that is named `processProperty1`, you can access that property with the following code:

```
properties.get("processProperty1")
```

Precondition scripts in component processes and generic processes can access the output properties of previous steps. For example, assume that the post-processing script of a step that is named `Step1` specifies a property that is named `OutputProp1`. The following precondition script allows another step to run only if the value of this property is `value1`:

```
properties.get("Step1/OutputProp1") == "value1"
```

Preconditions have access to a property that is named `hasFailures`. This property is `True` if any of the steps in the process fail. It is undefined if no steps fail. For example, to run a step only if no other steps in the process fail, use the following precondition:

```
properties.get('hasFailures') == undefined
```

Adding deployment warnings

Warning statuses provide a way to easily identify deployments that require closer inspection that had been marked during the process execution.

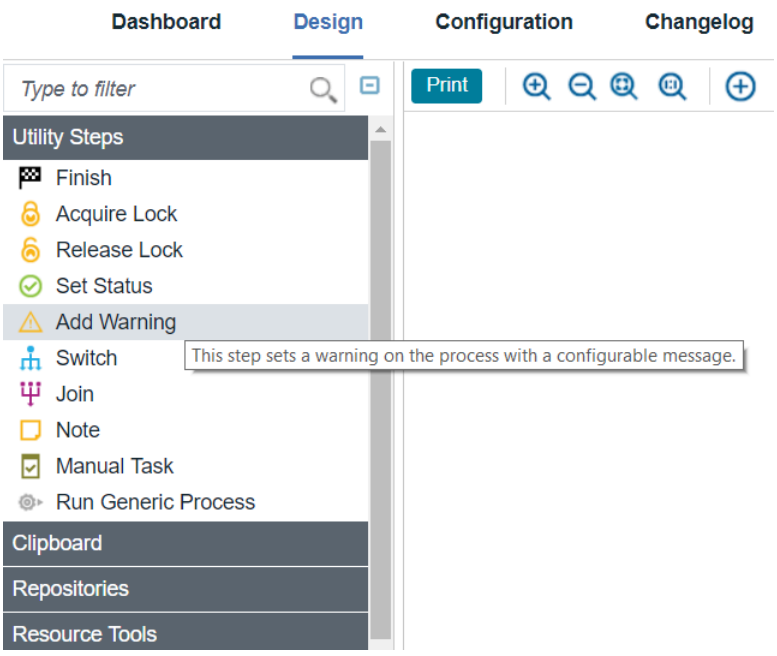
Adding warnings

The warning message will be determined by the process author at design time.

Warnings can be set on an execution by either of the following:

1. Through the `Add Warning` plug-in step.

   This step requires a warning message as its only field. The warning message has a minimum length of 1 and maximum length of 2000. Empty string messages will be discarded.



2. In the post processing script.

The default post-processing script is an example of how this is done:

```
// post-processing script
var exit = properties.get('exitCode');


/* Set a warning
scanner.register('^Warning: ', function(lineNumber, line) {
    properties.put('ucdWarning', line);
});

scanner.scan();
*/


if (exit == 0) {
    properties.put('Status', 'Success');
}
else {
    properties.put('Status', 'Failure');
}
```

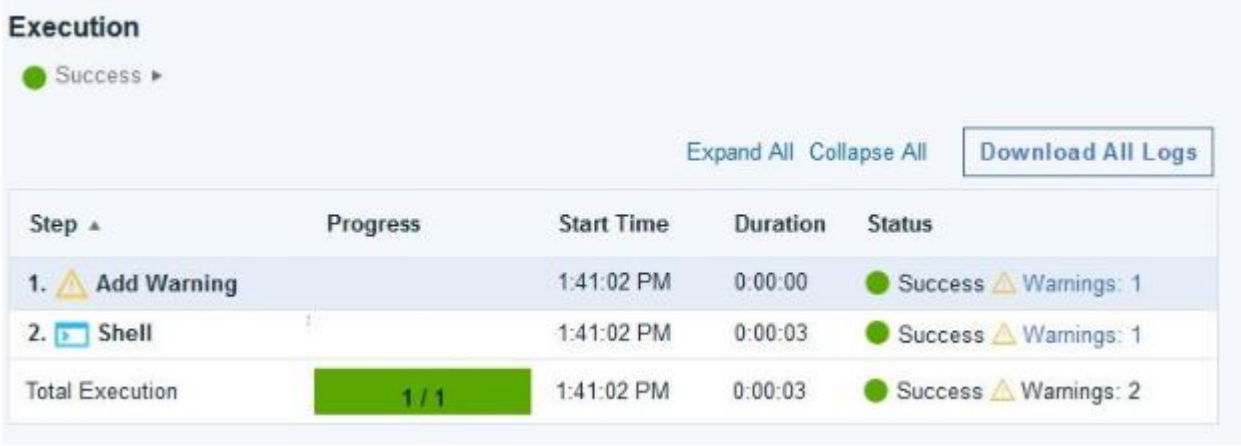
View process requests with warnings

Scan the request history to locate executions that have warnings set and easily trace the warnings back to their originating plug-in step. The **history** and **execution log** pages will display the status of the execution alongside a warning indicator with count of warnings. This indicator will only display if there are any warnings from child activities that have not been ignored. Users can click warning count to view the list of warning messages from the plug-in steps of the executed process activity.

The **history** page will display a `Status` column that shows the execution status (Success or Failure) and also the count of warnings that had been set for that process. No count will be given if no warnings were set.

The **execution log** page displays each step and child execution. When the steps have completed the Status column will display the count of warnings in the same manner as the **Request History** page. In addition to displaying how many warning messages were set on a step/execution, you can bring up the warning messages for that step and any child steps underneath it. Below is an example of what messages would be available at each step in the **execution log** page. The `shell with warning` steps set a warning message.

Figure 6. Execution log showing warnings



Steps	Warning Messages
(application process (execution))	"warningA, warningB"
1. Install Component "myComponent"	"warningA, warningB"
__ i. myComponent	"warningA, warningB"
____ A. myComponentProcess	"warningA"
______ a. Shell with warning	"warningA"
______ b. Shell	""
____ B. myOtherComponentProcess	"warningB"
______ a. Shell with warning	"warningB"

Success/Failure notifications include warnings

In addition to displaying the result status, the default deployment or process notification templates include the total count of warnings that have not been ignored. You have access to the warning messages if you choose to modify the default notification templates. Server upgrades will not modify existing notification schemes.

Behavior

Warnings set on plug-in steps are passed up to parent process. For example, if you have a component process which uses the plug-in step `Add Warning` and this component process is used in an application process, the warning would be available in the application process **execution log** and **history**.

Warning messages are stored in the workflow trace. Each activity trace may contain a field `warningMessage`containing the warning message if one is set. Workflow metadata will contain a count of warnings generated for each execution context to allow accessing the warning count without loading the workflow trace.

Activities can optionally ignore warnings from its child activities. An activity that can spawn child activities can be configured to ignore the warnings from its child activities. The UI has an option to ignore warnings from child activities. If the activity is configured to ignore warnings then any warnings set from child activities should not be included in the spawning-activity's execution log and process request history.

For example, a component process `deployWithWarning` will always set a warning message. There is an application process `appProcess` that runs an `Install Component` step that runs `deployWithWarning` but ignores warnings. After execution completion, the execution log and deployment history will show `appProcess` and the `Install Component` step was successful with no warnings. However if the execution log is expanded, `deployWithWarning` will show successful with 1 warning.

Specific activities that have the option to ignore child warnings are:

- Run Generic Process
- Run Generic Process for Each Affected Resource
- Run Component Process
- Run Application Process

- Run Operational Process for Multiple Components
- For Each Agent
- For Each Tag
- Run Process for Each Version
- Install Multiple Components
- Install Component

Existing History and Execution Log

The existing process request history and execution logs are not affected as they will not have the new plug in step `Add Warning` and post-processing script actions.

Post-processing scripts

Post-processing scripts run after a step finishes. Typically, post-processing scripts ensure that expected results occurred.

Notes:

- Use JavaScript™ 1.7 to write post-processing scripts.
- In the script, do not use the `${p:component.myProperty}` notation. For example, to set a property named `myProp` whose value is included in the output log, enter this script:

```
var exit = properties.get('exitCode');

scanner.register("regex", function(lineNumber, line) {
     var thing = 'do stuff';
});
scanner.scan();

if (exit == 0) {
    properties.put('Status', 'Success');
}
else {
     properties.put('Status', 'Failure');
}
```

- The syntax for post-processing scripts is similar to the syntax for preconditions. See Process step preconditions on page 2411.

Steps in component processes and generic processes can have post-processing scripts, but steps in application processes cannot.

Post-processing scripts come from the following sources:

- Steps can have default post-processing scripts. These scripts are included with the plug-in. See Creating plug-ins on page 2561.
- You can override the default post-processing script for a step or provide a post-processing script for a step that does not have one. See Editing processes on page 2453.
- You can store post-processing scripts on the server for reuse. See Storing reusable post-processing scripts on page 2242.

Post-processing scripts must set the **Status** property to specify the status of the script. The script can specify any value for the **Status** property. For example, to specify that the script was a success, run the following command in the post-processing script:

```
properties.put("Status","Success");
```

See The post-processing element on page 2576 for examples and more information.

Storing reusable post-processing scripts

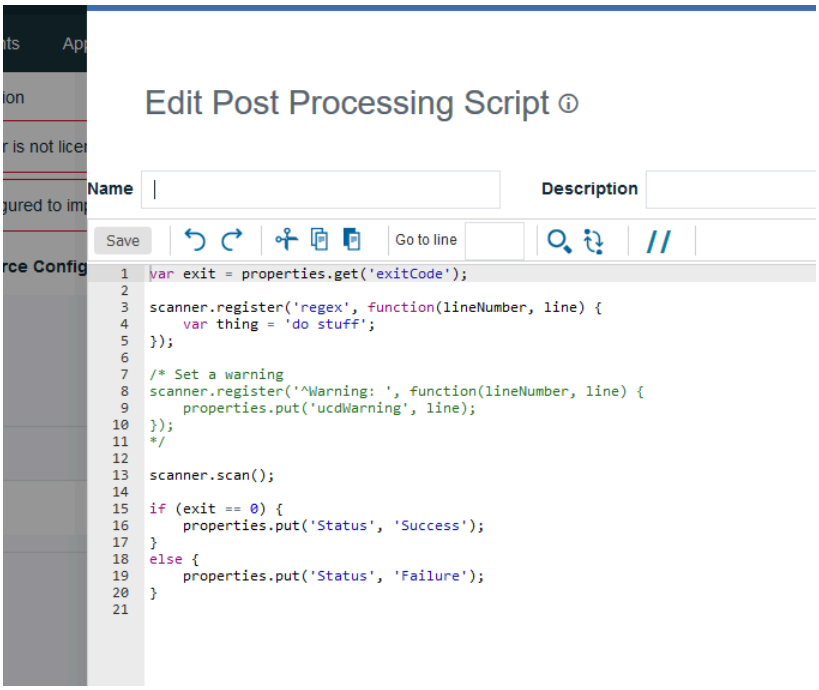
You can store post-processing scripts on the server so you can reuse them in component processes and in generic processes.

About this task

When a step finishes, the agent runs the post-processing script for that step. When the agent runs the script, it first loads the server log file. It finds the exit code property of the target step with regular expressions that are defined in the script. It then applies any actions that are defined in the script before it runs the next step.

The HCL DevOps Deploy (Deploy) server log file is normally found in the following location: *server_installation_directory*`/var/log/deployserver.out`.

1. Click **Settings > Post Processing Scripts** and then click **Create Script**.
   Result
   The **Edit Script** window opens, as shown in the following figure.



2. Specify a name for the process step.
3. Give a description for the process step.
4. Specify the teams that can access and edit the script.
5. Enter or paste the script into the **Script Body** field.
   See the roll-over help next to the field for information about the properties and variables that are available for user-defined scripts.

   Post-processing scripts must set the **Status** property to specify the status of the script. The script can specify any value for the **Status** property. For example, to specify that the script was a success, run the following command in the post-processing script:

   ```
   properties.put("Status","Success");
   ```

Examples of post-processing scripts

Post-processing scripts are written in JavaScript™.

Post-processing scripts must set the **Status** property to specify the status of the script. The script can specify any value for the **Status** property. For example, to specify that the script was a success, run the following command in the post-processing script:

```
properties.put("Status","Success");
```

Writing to the log

You can write to the output log for the process with the following command:

```
commandOut.print("hello");
```

Checking the result of a step

The post-processing script can examine the step's output log and run commands based on the result. In the following code fragment, `scanner.register()` registers the strings `error at line` and `the value is` with a regular expression engine, then runs commands if those strings are found. After all strings are registered, it calls `scanner.scan()` on the step's output log line by line.

```
properties.put("Status", "Success");

//
// Evaluate the built-in exitCode property, which indicates the exit code
// of the script called by the plug-in step. Typically, if the value of
// the exitCode property is non-zero, the plug-in step failed.
//
if (properties.get("exitCode") != 0) {
     properties.put("Status", "Failure");
}
else {

     //
     // Register a scanner to search for the text "error at line" in the log.
     // The first argument is a regular expression.
     //
     // The second argument, an inline function, is invoked once for
     // every line in the log output that matches the pattern. The "lineNumber"
     // variable contains the line number where the match occurred, and the
     // "line" variable is the full text of the line.
     //
     scanner.register("(?i)ERROR at line",  function(lineNumber, line) {

         //
         // In this case, we build up an "Error" property which
         // contains the text of all errors that are found. We find every
         // line starting with "error at line" and add it to this list.
         //
         errors.put("Errors", errorString);



         //
         // If a line starting with "error at line" is found, the step has
         // failed, so we set the special "Status" property to "Failure",
         // indicating to the DevOps Deploy server that the step should
         // be marked as a failure.
         //
         properties.put("Status", "Failure");
     });

     //
     // Multiple searches can be registered with the scanner. We add a
     // second search to look for some interesting text to set as an output
     // property.
     //
     // For example, if there is a line "The value is BLUE", then we end up
     // with an output property, "Value", with a value of "BLUE".
     //
     scanner.register("The value is", function(lineNumber, line) {
```

```
            var value = line.replace("The value is ", "");
            properties.put("Value", value);
        });

        scanner.scan();


        //
        // Convert the collected list of error strings into a single string and
        // set that as an output property.
        //
        var errors = properties.get("Error");
        if (errors == null) {
            errors = new java.util.ArrayList();
            properties.put("Error", errors);
        }
        properties.put("Error", errors.toString());
    }
```

Retrieving a property from a previous step

The following scripts assume two steps that are in the same process. The post-processing script for the first step creates two output properties, which are named **Status** and **Step1Prop**:

```
if (properties.get("exitCode")==0)
  { properties.put("Status","Success"); }
else
  { properties.put("Status","Failure"); }

properties.put("Step1Prop","value1");
```

Then, a second step can access those properties. For example, if the first step was named `Step1` and specified the output properties **Status** and **Step1Prop**, you might create a Shell step with the following code to access those properties:

```
echo Properties of previous step:
echo "${p:Step1/Status}"
echo "${p:Step1/Step1Prop}"
```

The second step can also use those properties in its post-processing script. For example, the following post-processing script includes the **Status** property of the previous step in a property for the second step:

```
{
properties.put("Step2Prop","New property");
properties.put("Status","OK");
properties.put("StatusOfPreviousStep","${p:Step1/Status}");
}
```

Passing a value from one step to another

Passing a value from one step to another involves a few general steps:

1. In the first step, store the value in a place where the post-processing script can access it. You can store the value by writing it to the log file for the step. For example, if you are using a Shell step, you can write the value and a unique key to the log with an `echo` command:

   ```
   echo "myvalue=1234"
   ```

   As a result, the log for the step shows the key and value:

   ```
   myvalue=12345
   ```

2. In the post-processing script for the step, use the scanner object to scan the log for the key. Then, assign the value to an output parameter. The following example script searches the output log for the key `myvalue=` and then stores the value in the output property `myOutputValue`:

   ```
   var exit = properties.get('exitCode');

   scanner.register("my value=", function(lineNumber, line) {
        var value=line.replace("my value=","");
        properties.put("myOutputValue",value);
   });

   if (exit == 0) {
       properties.put('Status', 'Success');
   }
   else {
        properties.put('Status', 'Failure');
   }
   scanner.scan();
   ```

3. In future steps, refer to the output parameter of the first step. For example, if the first step is named `SetMyValue`, future steps can refer to the property with the following code:

   ```
   ${p:SetMyValue/myOutputValue}
   ```

Adding a deployment warning

A warning can be added by a post-processing script at design time using the process author. Here's the default post-processing script which is an example of how this is done:

```
// post-processing script
var exit = properties.get('exitCode');



/* Set a warning
scanner.register('^Warning: ', function(lineNumber, line) {
    properties.put('ucdWarning', line);
});

scanner.scan();
*/



if (exit == 0) {
    properties.put('Status', 'Success');
}
else {
```

```
    properties.put('Status', 'Failure');
}
```

Allowlisted in-line functions and objects

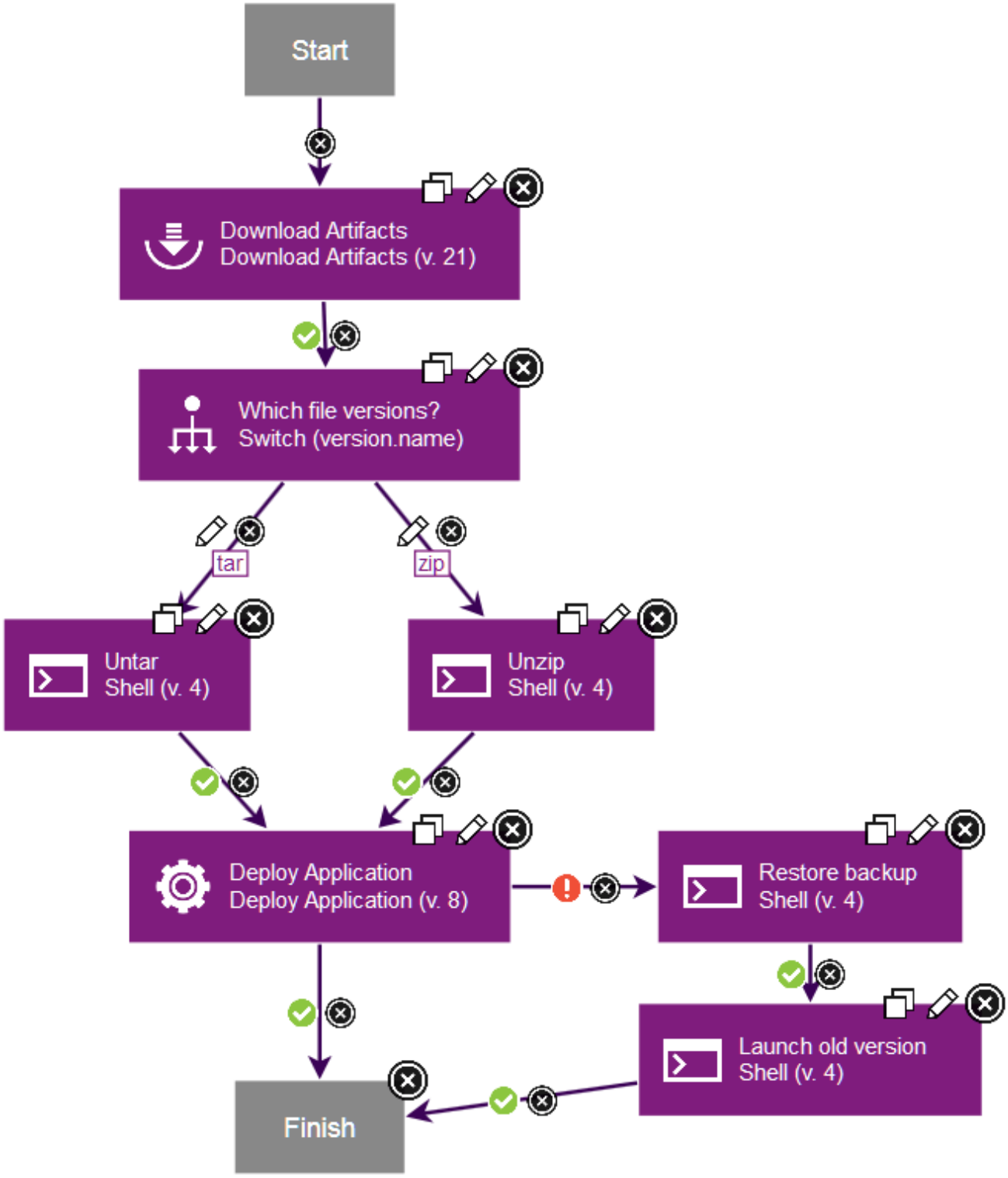
The following objects and functions are allowed to be used in the JavaScript runtime:

```
Function
Object
Error
decodeURI
decodeURIComponent
encodeURI
encodeURIComponent
escape
isNaN
unescape
NaN
isFinite
parseFloat
parseInt
Infinity
undefined
EvalError
RangeError
ReferenceError
SyntaxError
TypeError
URIError
Array
String
Boolean
Number
Date
Math
JSON
RegExp
```

Component processes

A *component process* is a series of user-defined steps that operate on a component's artifacts. Each component has at least one process that is defined for it and can have several.

A component process can be as simple as a single step or contain numerous steps and relationships. The *switch step*, for instance, creates conditional processes. For example, you might take artifacts from a source like an IBM® Anthill Pro project and map the artifacts that get deployed to an HTTP server into one component. Then, you might map the artifacts that get deployed to a Java™ Platform, Enterprise Edition container to another component. You might map the artifacts that get deployed to a database to yet another component. Or, to take another example, a single-component deployment might consist of two processes: the first moves component files to a target server on Friday night (a lengthy operation), while the second deploys the files Saturday morning.



Important: When you begin to create a component process, you must assign a type by making a selection from **Process Type** menu. The type that you select can determine whether the application process that you create later

functions as intended. For example, if the object of the component process you are creating is to deploy the current component and you select **Operational (with version)** in **Process Type**, the process will fail. See Component process types on page 2423.

Component processes are created with the *process editor*. The process editor is visual drag-and-drop editor where you can drag process steps onto the design space and configure them as you go. As more steps are placed, you visually define their relationships with one another. Process steps are selected from a menu of standard steps that replace typical deployment scripts and manual processes. HCL DevOps Deploy (Deploy) provides steps for several utility processes, such as inventory management, and workflow control. Other process steps are provided by plug-ins. Plug-ins provide integration with common deployment tools and application servers, such as WebSphere®, Microsoft™ IIS, and many others. Without any additions, Deploy provides plug-ins for many common processes, such as downloading and uploading artifacts, and retrieving environment information. A component process can have steps from more than one plug-in.

Like other processes, component processes begin with the Start step and must end with the Finish step.

A component process is defined for a specific component. A component can have more than one process that is defined for it, but each component requires at least one process.

For example, deploying an EAR file to an IBM® WebSphere® Application Server typically consists of the following operations:

1. Transfer the EAR file to the target server.
2. Stop the target server instance.
3. Run wsAdmin with deployment properties.
4. Start the target server instance.
5. Verify that the deployment succeeded by accessing a specified URL.

The WebSphere® Application Server - Deploy plug-in provides a configurable process step for each operation.

A frequently used component process can be saved as a template and applied later to new components.

Component processes are run by Deploy agents on hosts. One instance of a component process is started for each resource that is mapped to a component in the target environment. See Resources on page 2312.

Component process types

Different types of component processes deploy, configure, or uninstall component versions.

In most cases, you run component processes by adding them to application processes. However, you can also run component processes directly. For more information, see Running component processes on page 2427.

The Deployment and Operational (With Version) process types require a component version. If you do not specify a component version, the process does not run. In this case, the process is shown in the log with this message: `No Version Selected`. You can access the component version name with the **${p:version.name}** property. See Referring to properties on page 2509.

The other component process types do not use a specific component version; they run the same way regardless of the inventory of the target resource. Therefore, version-specific properties such as the **${p:version.name}** property are not available.

You use the following component process types:

Deployment

This type of component process deploys the specified component version to the target resource. It is the most common type of component process. If no component version is specified, the process does not run. For example, if the deployment manifest or snapshot does not contain a version of the specified component, the process does not run.

After the process runs, it updates the component inventory to include the component version, even if the process failed. If the process finishes, it updates both the component inventory and the configuration inventory, which contains all the properties for that component and its related elements. To view the updated properties, see Viewing resource configuration inventories on page 2397.

In most cases, this type of component process includes the Download Artifacts step, which downloads the component version artifacts to the target resource. Then, the process includes steps that install or configure the component version.

The most common way to run a deployment component process is to add it to an application process. In an application process, add the Install Component step, and then select the deployment component process.

Configuration Deployment

This type of component process applies a configuration inventory to a deployed component.

After the process runs, it updates the resource configuration inventory to contain updated properties. It does not change the environment or component version inventories.

For example, this type of process might use configuration templates to create configuration files, as described in Creating configuration files in component processes on page 2427. As another example, this type of process can update an application server with the most recent component properties, environment properties, and application properties. In this case, the process updates the configuration on the application server without reinstalling the component.

To include this type of process in an application process, add the Apply Configuration step, and select the configuration deployment process.

Uninstall

This type of component process removes the component version from the target resource. It also removes the component version from the environment inventory. It does not modify the configuration inventory.

To use this type of process in an application process, add the Uninstall Component step, and select the Uninstall component process.

This type of process can uninstall all the component versions in the inventory or only the specified versions. When you add a component with this component process to an application process, you specify the **Uninstall Type** field. The **All Existing** option removes all the versions of this component that are in the inventory. The **All Selected For Process** option removes only the versions that you specify when you start the application process.

Operational (With Version)

This type of process does not add or remove artifacts from the inventory; it runs steps on a currently deployed component. For example, this type of process can start or stop a service or change the properties on a deployed component.

By default, this type of process always runs when you include it in an application process, but you can restrict it to run only when certain components change.

To use this type of process in an application process, add a step that can call this type of process. For example, the Run Process for Each Version and Install Component steps can call this type of process.

Running this process step does not update the component's properties on the target resource.

Operational (No Version Needed)

This type of process is the same as the Operational (With Version) type, except that this type does not use a component version.

Each process of this type is listed as an available application process step, and you can add its step to call the process. You can also add a step that can call this type of process, such as the Apply Configuration step, to use this type of component process in an application process.

Creating component processes

A component process is a series of user-defined steps that operate on a component's artifacts. You create a component process in two steps: first, you configure basic information, such as name; second, you use the process editor to assemble the process.

1. Log in to the HCL DevOps Deploy (Deploy) application.
2. Click **Components > *selected component* > Processes > Create Process**.
   The **Create Process** window is displayed.
3. Provide the process details.

Field	Importance	Description
Name	Required	Enter a name (maximum 255 characters) that identifies the process. The process name appears in many UI elements.
Description	Optional	Conveys more information about the process.
Process Type	Required	Defines the process type.

Field	Importance	Description
		Selecting the correct process type is critical for the associated application process to run correctly. **Important:** If you are deploying components you must select the **Deployment** process type. If another type is selected for a component that you plan to deploy with an application process, the application process will fail. For more information, see Component process types on page 2423.
Inventory Status	Required	Status that is applied to component versions after being successfully run by this process. `Active` indicates that the component version is deployed to its target resource. The status appears on the **Inventory** panes for the component itself and environments that ran the process.
Default Working Directory	Required	Defines the location that the agent uses to run the process (for temporary files, and so on). The default value is `${p:resource/work.dir}/${p:component.name}`, where `${p:resource/work.dir}` is the default working directory for the agent and `${p:component.name}` is the component name. The default value resolves to *`agent_directory`*`\work\`*`component_name_directory`* and is stored in the `componentProcess.defaultWorkDir` property. The default properties work for most components; you might need to change it if a component process cannot be run at the agent's location.
Required Role	Optional	Restricts who can run the process. You can choose multiple roles associated with the team for the environment to run the process. The available options are derived from the Deploy security system. The default value is `None`, meaning anyone can run the process. For information about security roles, see Managing Security on page 2149.

4. Save the process details.

   Result

   The process is listed on the **Processes** pane for the associated component.

What to do next

You can edit the process in the process editor. See Editing processes on page 2453.

Running component processes

Component processes are typically run from within application processes. However, you can also run component processes directly to test the processes or to correct an error in an urgent situation.

About this task

To run a component process in an application process, add the appropriate step to the application process and specify the component process. For example, to run a deployment component process, use the Install Component step. The step that you use depends on the type of component process. For more information, see Component process types on page 2423.

You can run certain types of component process from generic processes. See Generic processes on page 2451.

You can also run component processes directly as follows:

1. Click an application, and go to the **Components** tab.
2. Next to the component, click **Run Process**.
3. In the **Environment** and **Resource** lists, select where to run the process.
4. In the **Process** list, select the component process.
5. If the component process requires a component version, in the **Version** list, select the component version.
6. If the component process requires other properties, specify values for those properties.
7. Click **Submit**.

Results

The component process is listed in the **Components > *selected component* > Component Request History**.

What to do next

You can search a component process by filtering any of the columns from the **Component Request History**:

- Resource
- Version
- Application
- Environment
- Scheduled For
- By
- Status

Creating configuration files in component processes

Configuration templates are templates for files that are created during a component process. Typically, the data is for configurations for web servers, but the files can be used for any application that you deploy.

About this task

A configuration template contains property names and property references. For example, the following line can be used in a configuration template:

```
database.username=@DB_USER@
```

When you deploy the component in an environment, the value of the **DB_USER** property that is defined for that environment is substituted for the `@DB_USER@` reference.

As an alternative to using configuration templates, you can also use the Create File step. In this case, you specify the contents of the file in the **Contents** field and include property references in the usual syntax of `${scope/propertyname}`, where `scope` is the scope of the property and `propertyname` is the name of the property.

1. Click **Components > *selected_component* > Configuration > Configuration File Templates > Create Configuration Template** to open the **Create Configuration Template** window.
2. Type a name in the **Name** field.
3. In the **Template** field, enter or paste the template text.
   Text can be in any script. The amount of text is based on the database that HCL DevOps Deploy (Deploy) uses.
4. Save the configuration template.
5. To use the configuration template in a process, add the Install Template step to a component process:

    a. In a component process, add the Install Template step.
    This step creates a file that is based on the configuration template.

    b. In the step properties, specify the name of the configuration template.

    c. In the step properties, select the **Show Hidden Properties** check box.

    d. In the **Environment Property List** field, specify the properties to bring into scope.
    Example
    For example, to bring only the environment properties into scope, specify `${p:environment/allProperties}`. In this case, the configuration template can refer to only environment properties. To bring environment properties, agent properties, and resource properties into scope, specify the following value for the **Environment Property List** field:

    ```
    ${p:environment/allProperties}, ${agent/allProperties}, ${resource/allProperties}
    ```

    e. Save the step and the process

Results

When you run the component process, the Install Template step creates a file that is based on the configuration template. The step replaces the property references with the property values. The default name of the file is the same as the name of the configuration template.

Application processes

You can use application processes to deploy or to roll back components.

HCL DevOps Deploy (Deploy) provides several common process steps. Otherwise, application processes are assembled from processes that are defined for their associated components. Application processes, like component processes, are created with the process editor.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes, the application determines which ones run and in which order. For instance, an **n**-tiered application might have a web tier, a middleware tier, and a database tier. And, continuing the example, the database tier must be updated before the other two, which are then deployed concurrently. An application can orchestrate the entire process, including putting target servers online and offline for load balancing as required.

An application process is always associated with a target environment. When an application process runs, it interacts with a specific environment. At least one environment must be associated with the application before the process can run. Application processes are independent of environment; processes can be designed independently of any particular environment. These application processes enable a single application to interact with separate environments, such as QA, or production. To use the same application process with multiple environments (a typical scenario), you associate each environment with the application and run the process separately for each one.

In addition to deployments, several other common processes are available, such as rolling-back deployments. Deploy tracks the history of each component version, which enables application processes to restore environments to any point.

Application processes and the steps that they comprise are configured with the process editor. For information about using the process editor, see Processes on page 2398. For information about individual process steps, see Process steps: Reference on page 2466.

Creating application processes

You can use application processes to deploy or to roll back components.

1. Display the **Create an Application Process** dialog box (**Applications > *select application* > Processes > Create Process**), and enter the following information:

Field	Description
Name and **Description**	Typically the name and description correspond to the application you plan on deploying. **Note:** You must specify a unique name. Your application process name cannot contain the same sequence of alphanumeric characters with different capitalization as an existing application process.
Inventory Management	If you want to handle inventory manually, select `Advanced`. To have inventory that is handled automatically, leave the default value, `Automatic`, selected. See Inventories on page 2394.

Field	Description
Offline Agent Handling	Specify how the process reacts if expected agents are offline: ◦ **Check Before Execution**: checks to see whether expected agents are online before you start the process. If agents are offline, the process does not run. ◦ **Use All Available**; **Report Failure**: process runs while at least one agent is online; reports any failed deployments because of offline agents. Useful for rollbacks or configuration deployments. ◦ **Always Report Success**: process runs while at least one agent is online; reports successful deployments.
Required Role	Use this list to select the role that a user must have to run the process. You will need to be in the required role for the team on the environment you are trying to run the process on. For more information, see Roles and permissions on page 2187. The default value is `None`.
Disable Deploying Snapshots	Allows you to create an application process that deploys component versions only. On enabling the option, you are forced to specify the component versions while deploying applications and alerts you that snapshot deployments are disabled.

2. Save your work.

What to do next

You can edit the process in the process editor. See Editing processes on page 2453.

Deploying components in an application process

You can deploy one or more components in an application process by calling component processes.

Before you begin

- Create an application and add one or more components to the application.
- Create an application process.
- Create a component process of the Deployment type for each component.

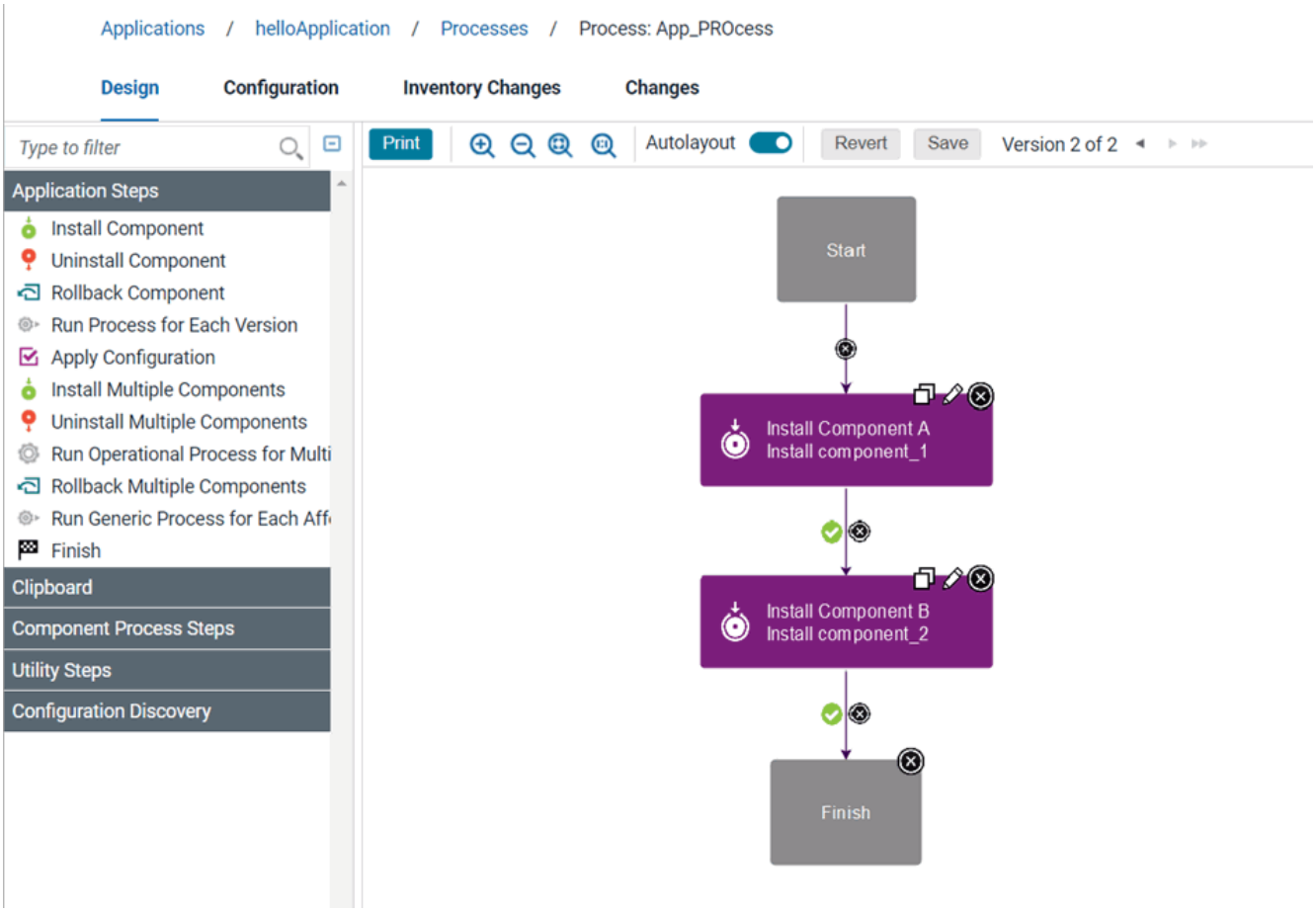
Important: If **Deployment** is not selected for **Type**, the deployment process will not succeed.

1. In an application process, add the Install Component step to the process, and click the edit icon . The **Edit Properties** window opens.
2. In the properties for this step, specify the component to deploy and the component process to run.
   For more information about this step, see Install Component on page 2476.
3. When you drag more steps to the process diagram, position the step so that the link between steps is highlighted, and then drop the step. The step is automatically linked to the earlier and subsequent steps.
4. Verify that all processes are connected in the correct order and that the flow is connected to the Start and Finish nodes; then, save the process.

5. Run the process on an environment.
   See Running application processes on page 2431.

Example

The following example application process deploys two components. The `Install Component A` and `Install Component B` steps call the deployment processes for each component.



Running application processes

To run an application process, specify the environment to use and the process to run. In many cases, you also specify the component versions to deploy.

Before you begin

You must have completed the following tasks:

- Created an environment.
- Created an application process.

About this task

Run an application process with a specific component version.

1. Click **Applications**, click the application, and then next to the environment, click **Request Process**.
   The **Create Deployment** wizard opens.
2. Set the process details.

   a. **Optional:** Re-select the application and environment names, if required.
   The **Application** and **Environment** fields are auto-filled based on the selection made in step 1.

   b. **Optional:** Enter the description for the process in the **Description** field.

   Note: You can also request a parameter. For example,

   ${**p:request.user.name**} (${**p:request.user.id**}) from ${**p:request.id**}

   c. To skip component versions that have not changed since the previous deployment, enable the **Only Changed Versions** option.
   If you do not select this option, all of the component versions that you select are deployed, even if they are already in the inventory.

   d. Select the application process from the **Process** list.

3. Choose the component versions or snapshot to deploy.
   If the process you selected deploys component versions, you must specify the component versions to deploy. You can use select a snapshot of component versions, or multiple component versions manually.
   Choose from:
   - To use the component versions in a snapshot, specify the snapshot from the **Select a Snapshot** list.
   - Specify component versions manually from the **Choose Component Version** window. You can add the component versions using any one of these actions:
     - To specify the versions for each component individually, click **Add** next to a component in the table, and select versions to deploy from **By Version Lookup**, and then choose any one of the following criteria:

Option	Description
Latest Available	Selects the most recent component version at the time that you make this selection regardless of when the process runs. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that is available then, click **Latest Available at Execution Time**.
Latest with Status	Selects the most recent component version at the time that you make this selection. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that has the specified status that is available at the later time, click **Latest with Status at Execution Time**.

Option	Description
Latest Available at Execution Time	Selects the latest component version that is available when the process starts. The version that is used might be generated after you make this selection.
Latest with Status at Execution Time	Selects component that have a status that you specify in the next dialog box. The version that is used might be generated after you make this selection.
Current Environment Inventory at Execution Time	Uses the environment inventory that is available when the process starts. The environment inventory might be different from the inventory that is available when you make this selection.

- To specify versions for all the components at one time, click **Select For All**, and then click one of the following options:

Option	Description
Latest Available	Select the most recent component version at the time that you make this selection regardless of when the process runs. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that is available then, click **Latest Available at Execution Time**.
Latest with Status	Select the most recent component version at the time that you make this selection. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that has the specified status that is available at the later time, click **Latest with Status at Execution Time**.
Versions With Name	Select the most recent component version whose name matches a string.
Current Environment Inventory	Uses the versions that are listed for that environment when you select it. This selection is typically related to the last version that was deployed to a certain environment of each component. For example, say that you deployed component A, version 1, to your environment. Then, in a separate deployment, you deployed an incremental version 1.1 to the same environment. If you look at the current environment inventory in the **Application** view, you see that the environment contains versions 1 and 1.1. Typically, you don't want to redeploy what is already there. However, this selection is useful when a deployment fails or is modified on the target computer itself, outside of Deploy. So if the deployment of version 1.1 succeeded on 10 agents but failed on two,

Option	Description
	you can schedule essentially a retry. Use of this selection is not common.
Latest Available at Execution Time	select the latest component version that is available when the process starts. The version that is used might be generated after you make this selection.
Latest with Status at Execution Time	Selects component that have a status that you specify in the next dialog box. The version might be generated after you make this selection.
Current Environment Inventory at Execution Time	Uses the environment inventory that is available when the process starts. The environment inventory might be different from the inventory that is available when you make this selection.
None (Clear All)	Clears your selection.

- To filter the list of components to the components with new versions, select the **Show only changed components** check box.

Notes:

- If you select multiple versions for one component, the versions are installed in the order you select.
- Make sure to select a version for each component. If you do not select a version for a component, that component is not included in the application process.

4. If the process has properties, specify values for those properties.
5. **Optional:** Schedule the deployment for a later date by disabling **Run Now** and then specify the date, time, and recurrence pattern of the deployment.
   You can choose the **Hourly**, **Daily**, **Weekly**, **Monthly**, and **Weekdays** recurrence pattern.
6. Review the deployment settings and click **Submit Deployment**.

Results

The server runs the application process with the specified component versions. You can view the process requests by going to the **History** tab for the environment or application. Typically, resources are mapped for each component version that is deployed by the application process. You can use the same application process for different environments, some of which do not have mappings for all components. If you run an application process on an environment where not all components are mapped, the resulting status for those components is `Not Mapped`.

What to do next

You can view the progress of the process by going to the **History** tab and opening the View Request. The application process request shows each step in the process. Within each process, the steps are not shown in the order that they run in. Instead, steps are shown in the order of appearance in the process editor, from the steps at the top of the canvas to the steps at the bottom of the canvas.

You can filter the **Application Process Request > Execution Log** based on the following **Status**:

- Any
- Not Executed
- Success
- Cancelled
- Failed

You can also search an application process by filtering any of the columns from the **History** tab:

- Environment
- Snapshot
- Scheduled For
- By
- Status

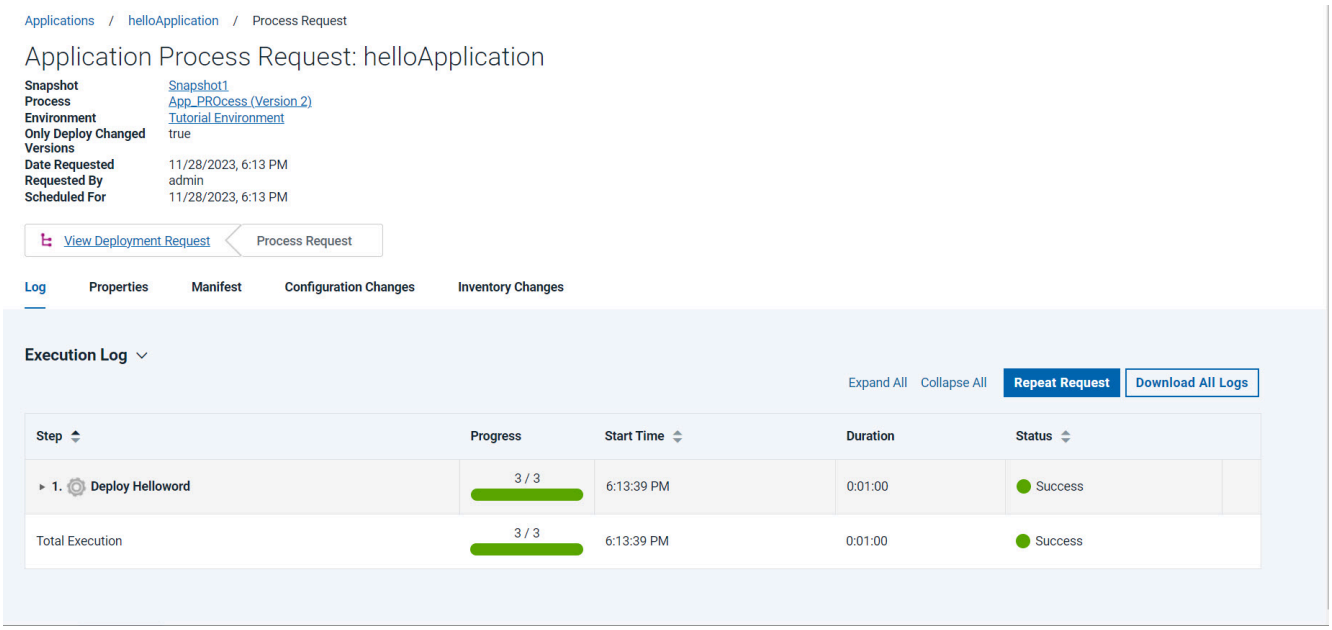
Related information

GitHub Action: Request Application process in Devops-Deploy

Running the same process again

The repeat request function is available after an application deployment completes. The **Repeat Request** feature is on the **Process Request** page for the application.

About this task



It repeats the same deployment that was previously requested, including the selected versions. Repeating a previous deployment might result in receiving an `Already Installed` message. When an application process step completes successfully, it is tagged with a version number. If a process does not complete successfully, it remains at the same version. Therefore, starting a repeat request function where some processes completed successfully and which some failed, a mixture of updated versions and remaining versions exist.

During the deployment process, the inventory of resources is verified. The resource version is compared to the requested deployment version. If the resource was successful in the previous build, the `Already Installed` message is displayed because the resource was successfully installed during the previous deployment process and it is not installed again.

Eliminating the `Already Installed` message, depends on how the request process is defined. For example, when a failed installation process results in mixed versions, you can manage environment inventory manually and remove artifacts for failed resources.

To start the deployment process without using the repeat request feature, start the request in the same manner as before and clear the **Only changed versions** options. Using this option causes the inventory check to be bypassed.

Uninstalling components with an application process

You can uninstall components by using an Uninstall Component step in an application process.

Before you begin

- Create an uninstall-type component process for each component version to remove. This process is run by the Uninstall Component step in an application process.
- If you intend to uninstall the component as part of a rollback process, create an application snapshot that contains the component versions to restore.

About this task

There are several ways to uninstall a component. To replace an existing component version, you can install another version over it by running an application process that contains a deployment process for the component. To remove a component version, you run an application process that includes the Uninstall Component step.

To uninstall a component version, complete these steps:

1. Create an application process that contains the Uninstall Component step.
2. In the Uninstall Component step, select the component to remove and the uninstall-type component process that is defined for it.
3. In the **Uninstall Type** field, specify **All Existing** to remove all versions of the component from the environment's inventory. Otherwise, specify **All Selected For Process** to remove only the versions that you select when you run the application process.
4. Run the application process.
   Result
   The selected component versions are no longer in the environment inventory.

Rolling back to a snapshot

You can roll back a component to the versions that are in a snapshot. In this case, you use a **Rollback Component** step with the **Remove Undesired Incremental Versions** option.

Before you begin

- Create a snapshot that includes the component versions to be restored. Snapshots are typically made for every completed deployment, especially for deployments in critical environments. See Creating snapshots on page 2383.
- Create an uninstall-type process for each component to be rolled back. For information about uninstalling component versions, see Uninstalling components with an application process on page 2436.

About this task

For a video demonstration of rolling back components, see Rollback Scenarios in Deploy.

When you roll back to a snapshot in this way, the server runs the uninstall process for each component version that is not in the snapshot. The server runs the uninstall processes in the reverse order in which it deployed the versions, starting with the most recently deployed version and ending with the version that was deployed first. Only the versions in the snapshot are left on the environment.

This method is intended to uninstall incremental versions, but you can also use it to uninstall full versions. However, for full versions, in many cases, it is simpler to install the new full version over the existing full and incremental versions.

This method works best for component versions that are already deployed, regardless of whether the deployments of those versions were completed. As an alternative, you can add a step that immediately rolls back deployments of versions that were not completed; for that method, see Rolling back components that fail to deploy on page 2439.

To roll back one or more components to the versions that are in a snapshot, complete these steps:

1. Create an application process with a name such as "Roll back to snapshot."
2. In the application process, add one or more **Rollback Component** steps, one for each component to roll back.
3. In the properties for each **Rollback Component** step, specify values for the fields that follow.

Tips:

- To specify these values, open the **Edit Properties** window by clicking the edit icon after you drag the steps to the process flow diagram.
- All steps in the process must be connected in a flow from the process Start node to the Finish node. When you drag the first step to the process, the step is automatically connected to the Start and Finish nodes. When you drag another steps to the process, position the step between two steps so

that the connection line is highlighted, and then drop the step. The step is automatically linked to the earlier and subsequent steps. If the connection line is not highlighted, you must manually make the connection from an earlier step to a subsequent step.

Component

Specify the component to roll back.

Component Process

Specify the uninstall process for the component.

Rollback Type

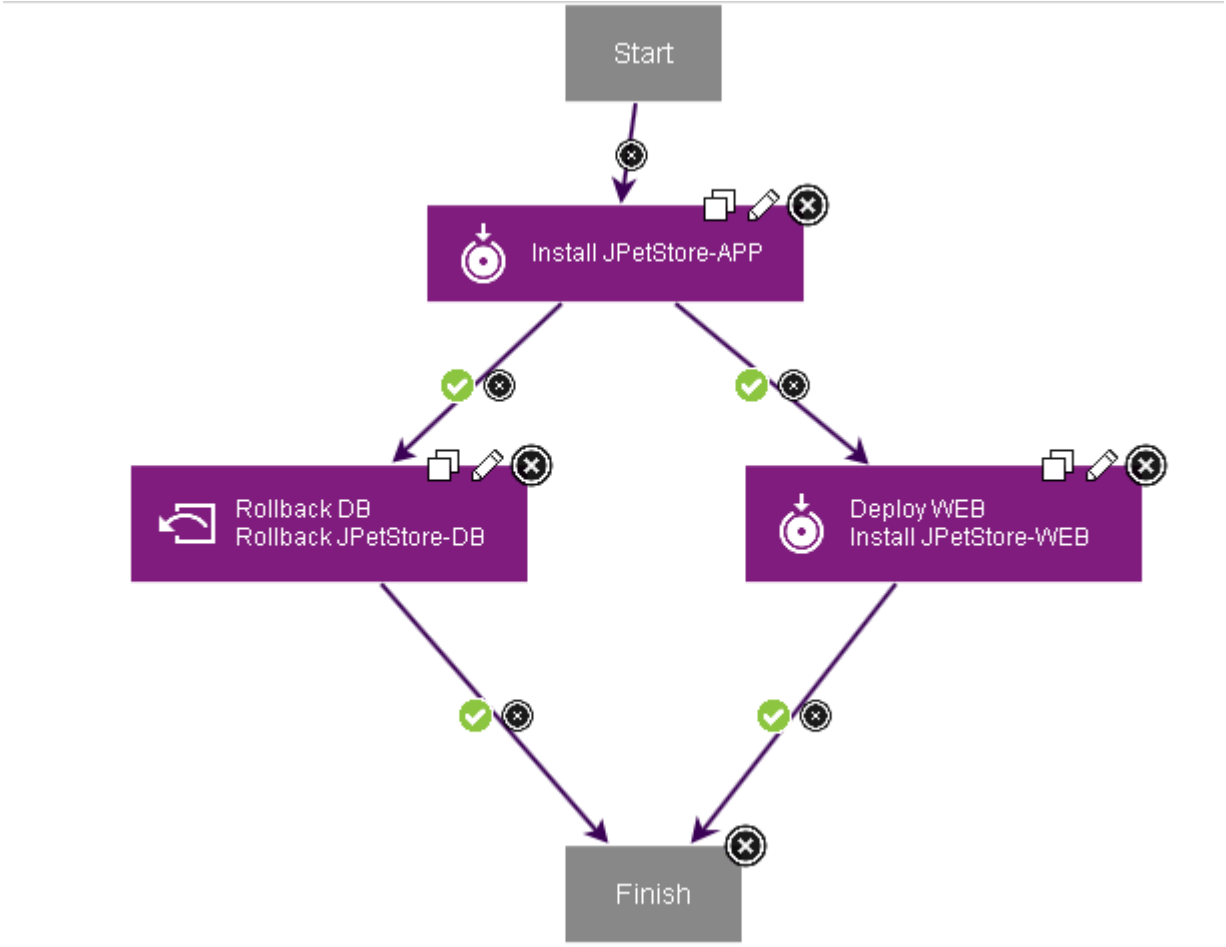
Select **Remove Undesired Incremental Versions**.

4. Verify that all steps in the process are connected in the order for them to run.

   To change the order of the steps, remove the links between steps by clicking the delete connection icon . Then, move the step into a new position and reconnect the steps in order.

   Example

   The following diagram shows a typical application process that includes a **Rollback Component** step. This application process is designed to roll back the three components in the application. Two components use an **Install Application** step that redeploys an earlier component version. The other component, the JPetStore-DB database component, uses a rollback step to roll back the incremental component version. This step uses the **Remove Undesired Incremental Versions** option and an uninstall-type component process.



5. Run the application process, and specify the snapshot to roll back to.
   If you are rolling back by redeploying an existing version, be sure to clear the **Only Changed Versions** check box. Otherwise, the process does not redeploy the version.

Results

The process uses the snapshot to determine which component versions to roll back. In each step that includes the **Remove Undesired Incremental Versions** option, versions that are not in the snapshot are removed.

Rolling back components that fail to deploy

If an application process fails to deploy a component, you can add a rollback step to the process to reverse the deployment. In this case, you use the **Replace with Last Deployed** option.

About this task

For a video demonstration of rolling back components, see Rollback Scenarios in Deploy.

To roll back a component in this way, you add a step in the application process that runs if the component is not deployed. This **Rollback Component** step with the **Replace with Last Deployed** option redeploys the previously installed component

version. The specified inventory still shows the new version and that the environment is noncompliant because the deployment of that version failed.

This method works best if you want to roll back a component within the same application process that attempted to deploy it. For example, you can install a component, run tests on it, and if the tests fail, roll back the component.

If the component is already installed and you want to switch to a different version, you can do one of these procedures:

- Install the new component version over the existing component version.
- Use the uninstall process for the component to remove the unwanted version.

  Create a rollback process that uses the **Remove Undesired Incremental Versions** option to roll back to a snapshot. See Rolling back to a snapshot on page 2437.

1. In an application process, add a step to deploy a component version as usual.
2. In the process, add a **Rollback Component** step.
3. In the **Rollback Component** step, specify the following information:

   Component

   Specify the component to roll back.

   Component process

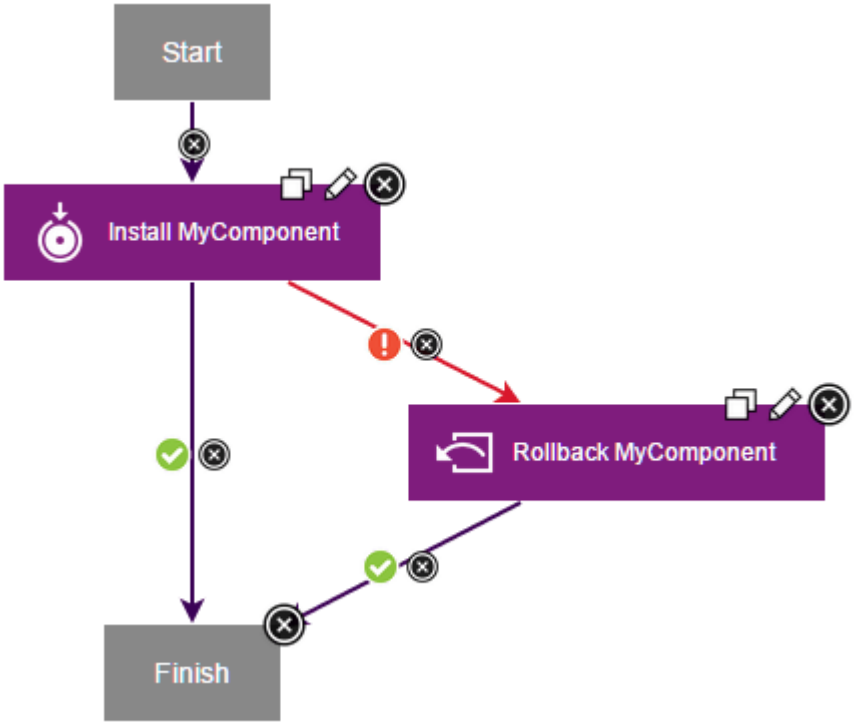
   Specify the process to run on the previously deployed version. In most cases, specify the deployment process for the component.

   Rolback Type

   Select **Replace with Last Deployed**.
4. Set a failure link from the deployment step to the rollback step so that the rollback step runs only if the deployment fails.

Example

Assume that version 1 of a component is deployed to an environment. You want to install version 2, but if that installation fails, you want to roll back to version 1. The application process looks like the following example. In this application process, the first step deploys the component version as usual. If that step fails, the rollback step runs with the **Replace with Last Deployed** option. In this case, the rollback step deploys version 1, which is the version that was in the inventory before the process started. However, the inventory is still noncompliant because you tried to deploy version 2 and failed.



Modeling blue-green, rolling, and canary deployments

The For Each Agent and For Each Tag steps provide ways to deploy to large environments in stages, allowing you to run flexible blue-green or rolling deployments, with or without updating canary nodes first.

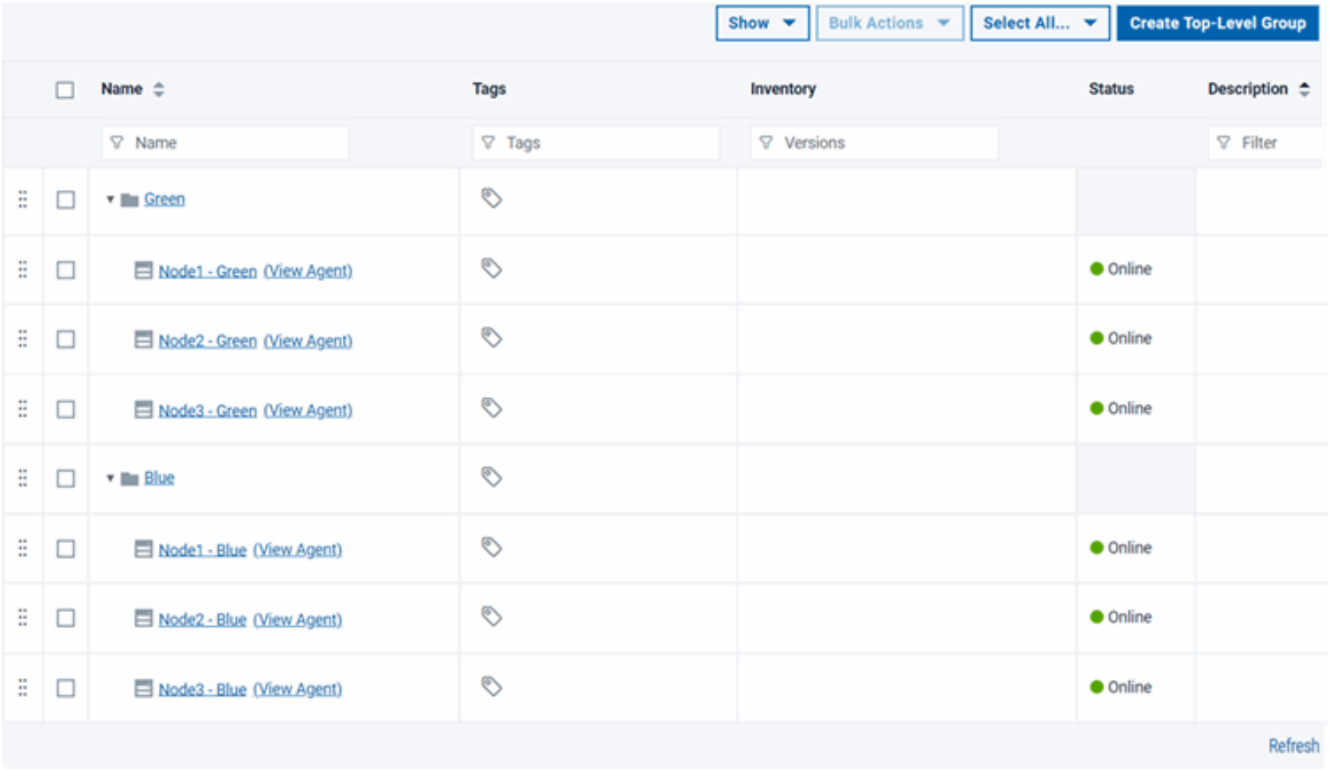
Two application process steps, the For Each Agent on page 2470 step and the For Each Tag on page 2472 step, provide flexible ways to deploy to multi-node environments in stages. You can use these steps to select subsets of agents and deploy to those subsets in specific orders. The following scenarios are just a few examples of how you can use these steps to control how deployments happen in multi-node environments.

Blue-green deployments

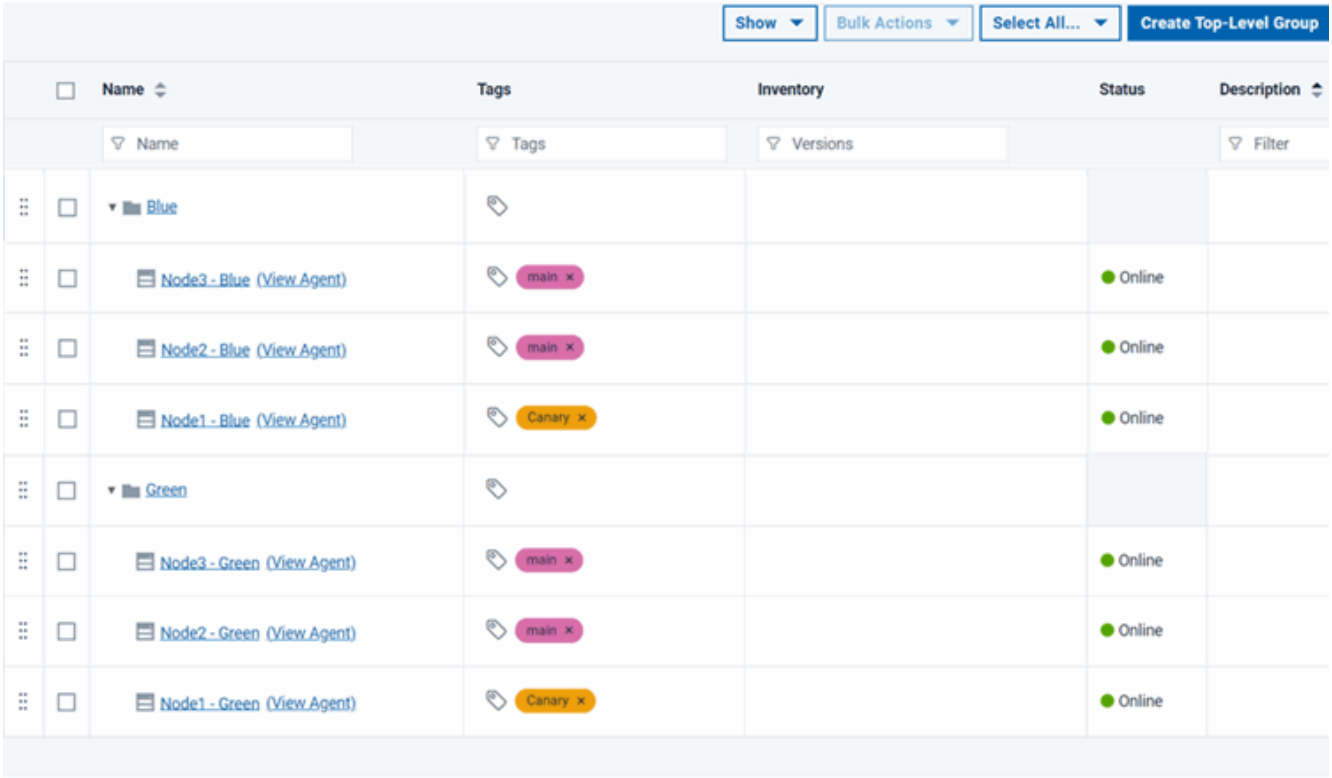
For a video demo of blue-green deployments, see Blue-green deployments with Deploy. For an article that covers a blue-green deployment scenario in detail, see Setting Up Blue-Green Deployments in Deploy.

Blue-green deployments use two identical environments. While clients are using one active environment, you can update the other environment without interrupting the active environment. Then, when the other environment is ready, you set a load balancer to send client traffic to the updated environment. Updating environments in this way allows for zero-downtime deployments and keeps a second working environment ready at all times in case anything goes wrong with the active environment.

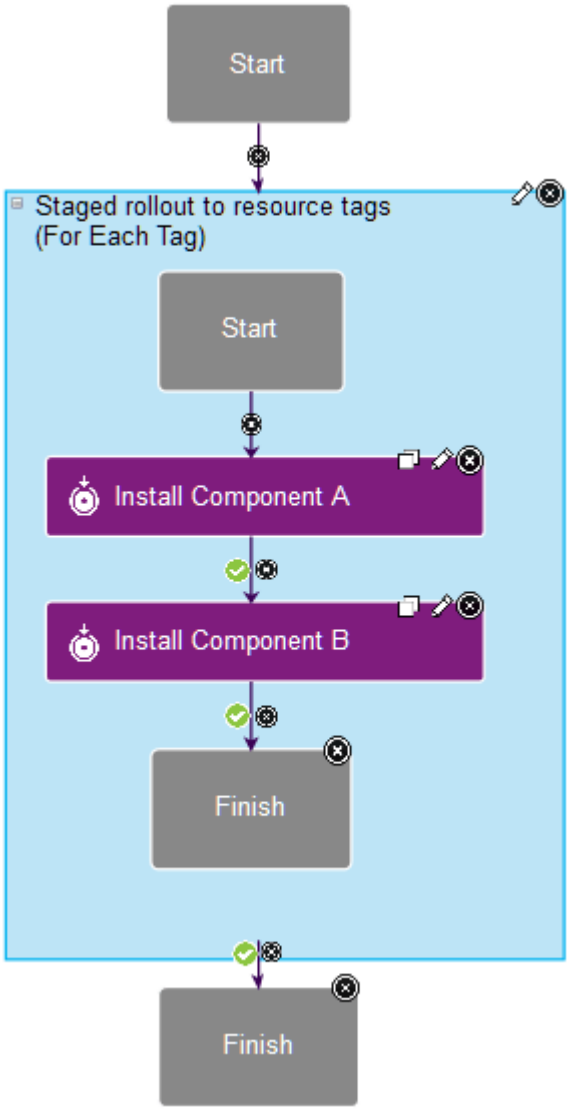
To set up a blue-green deployment, first set up matching environments on the server. These environments can point to the same physical environments as long as they have different agent resources. These different agent resources allow you to distinguish the two environments, as shown in the following figure.

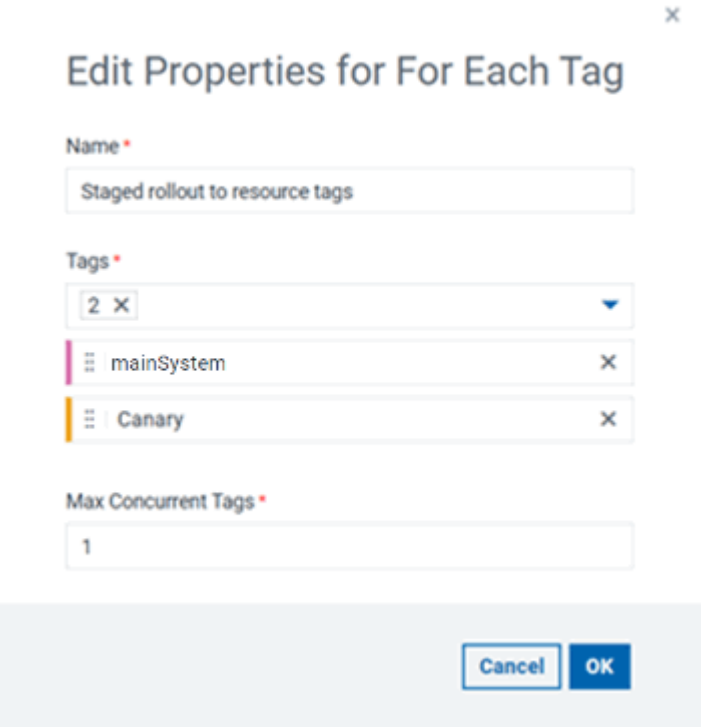


Then, apply resource tags to select subsets of the agent resources. For example, large deployments often update a few test nodes first, called *canary nodes*. You can apply a resource tag to your canary nodes to run deployments on those agent resources first. For example, the following figure shows the same resource tree with the addition of "canary" tags on the canary node and "main" tags on the other nodes:

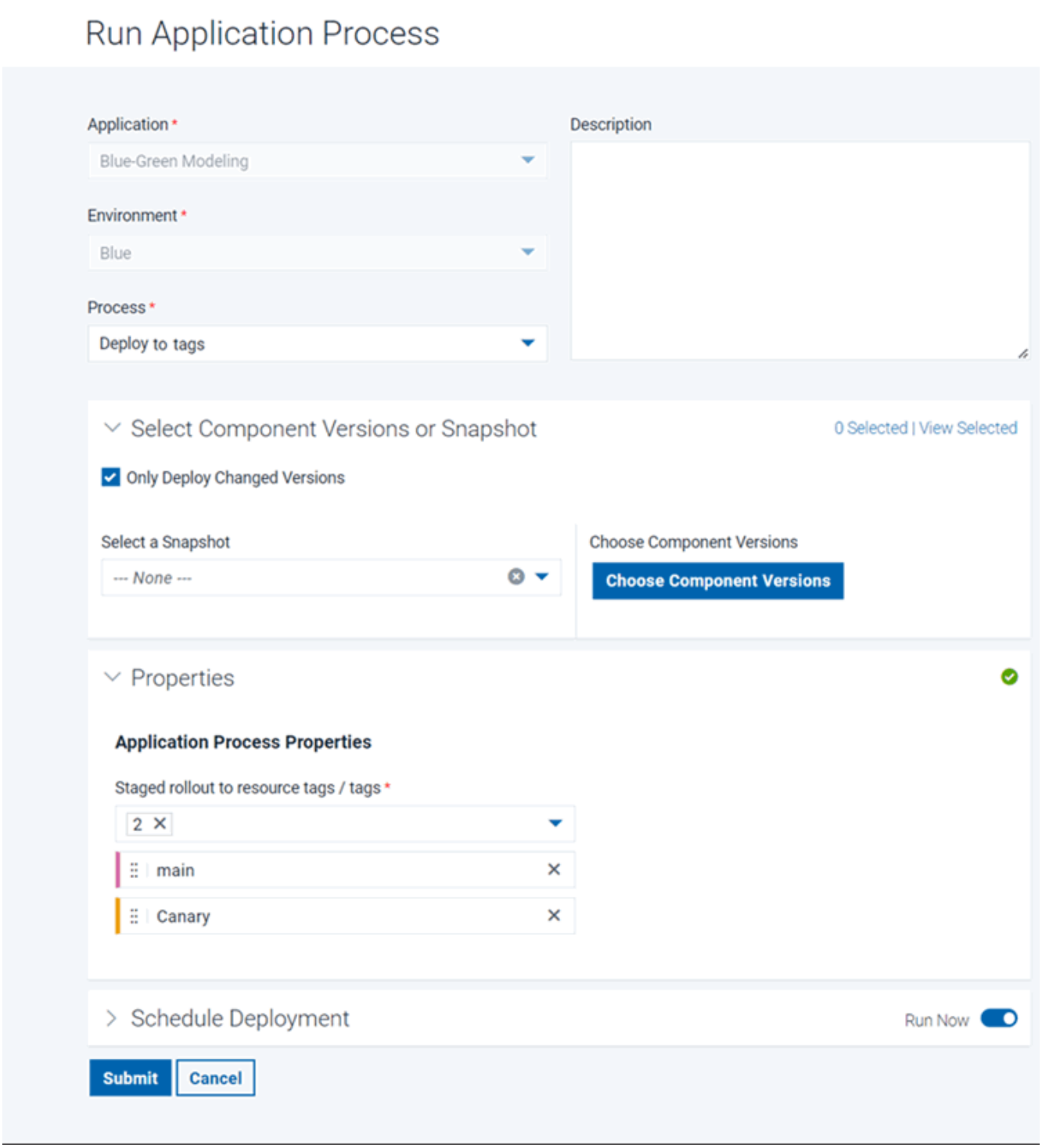


Then, in your application processes, run deployments based on the tags or agents. For example, to run a deployment on the canary agent resources first, and all of the other agent resources second, use a For Each Tag loop. In the loop, use the canary tag first and then a tag that represents the other resources. The following figure shows the For Each Tag step in use, and the next figure shows the properties for that step:


Start
Staged rollout to resource tags
(For Each Tag)
Start
Install Component A
Install Component B
Finish
Finish



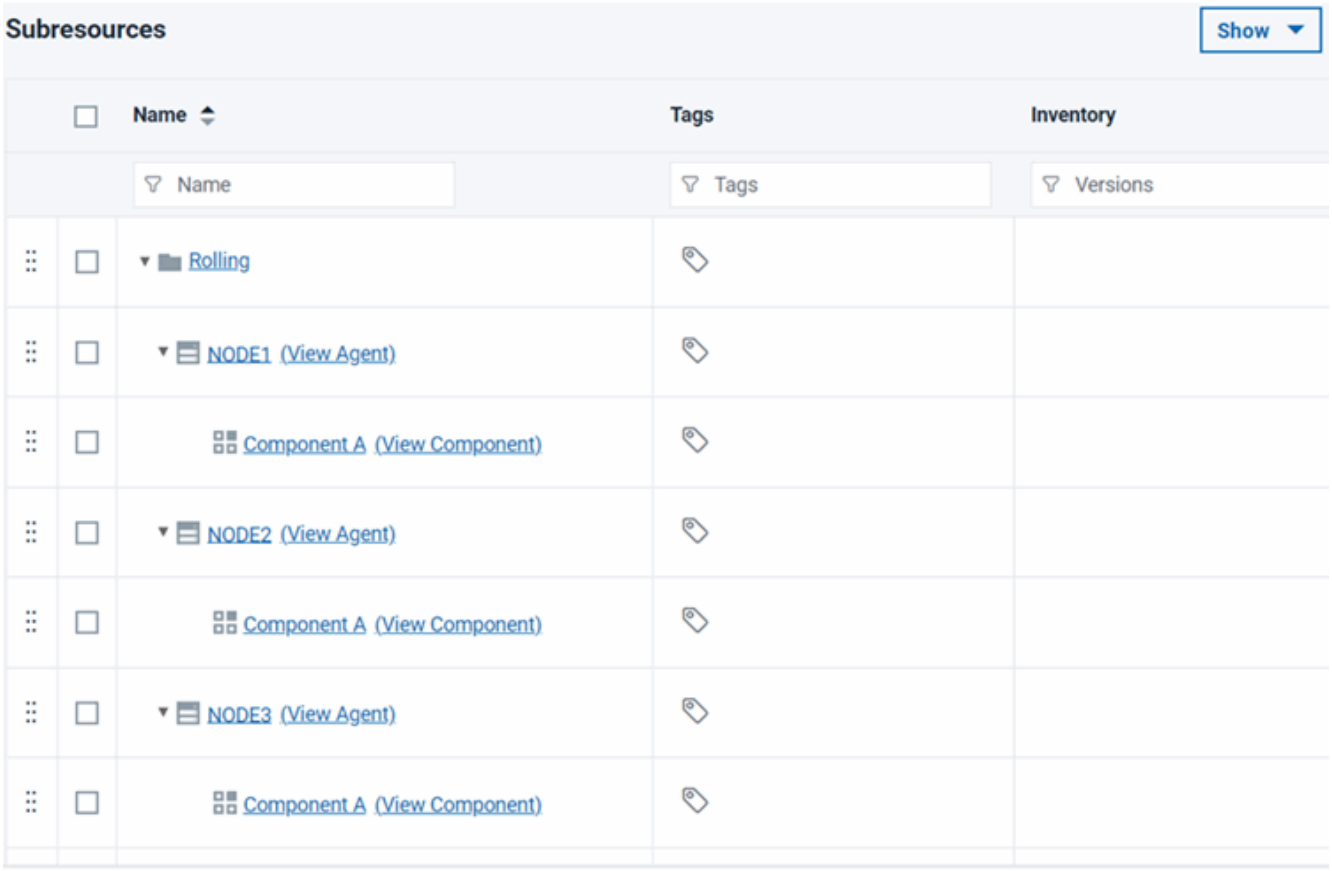
When you request the process, you can set the tags to use for that request, as shown in the following figure:



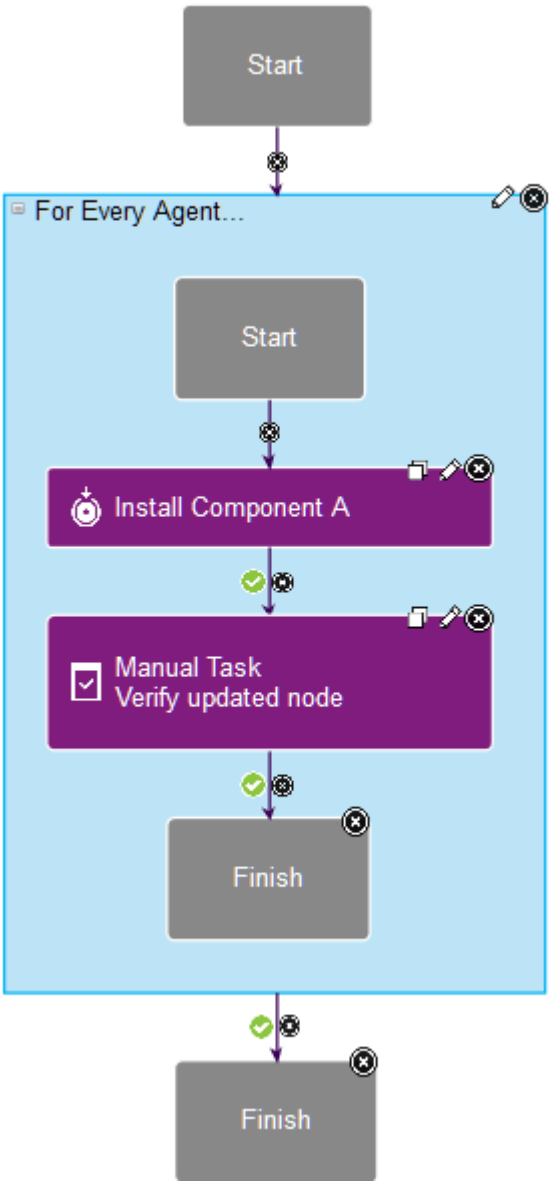
There are many other ways to organize this type of deployment. For example, instead of using separate environments, you could use a single environment and tag the nodes Blue and Green with resource tags. How you organize your environments and apply resource tags to them depends on the order in which you want to update your nodes.

Rolling deployments

Rolling deployments update large environments a few nodes at a time. The setup for rolling deployments can be similar to blue-green deployments, but this time with a single logical environment. The environment can have many nodes, as shown in the following figure:



To deploy to these nodes one at a time, use the For Each Agent step. For example, the following application process runs deployments on these nodes within the For Each Agent step. It pauses for a manual task after each deployment to ensure that the deployment ran correctly on that node. You could use any manual or automatic test in place of that manual task. In the properties for the loop, the **Max Concurrent Agents** property is set to 1 to run the deployment to a single agent at a time.



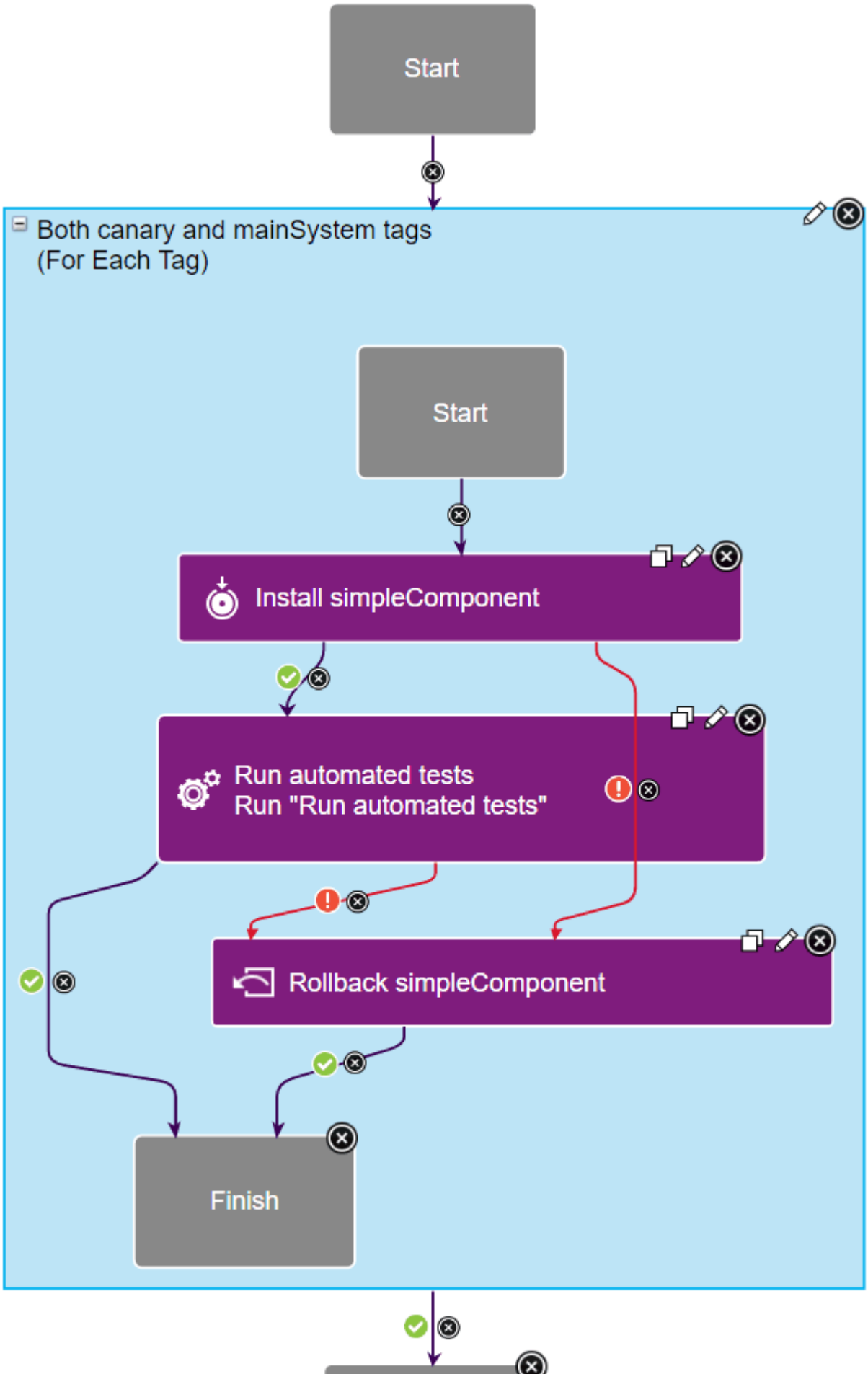
Canary deployments

Canary deployments are deployments that start by updating a single node or a subset of nodes. That way, you can test the update on a small subset of your system before you update every node. Canary deployments can be part of many other deployment scenarios. For example, the scenario in Setting Up Blue-Green Deployments in Deploy includes deployment to a canary node first.

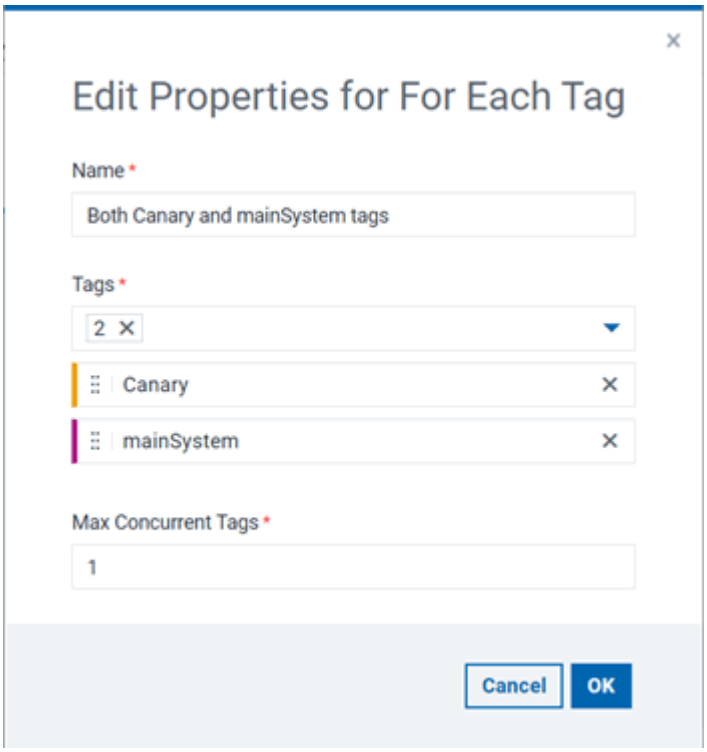
All you need to do to use a canary node in your deployment is to pick one or a few nodes and make sure that you deploy to those nodes first. Then, you must run automated or manual tests to make sure that the canary node or nodes are working well with the new version.

A straightforward way to deploy to canary nodes first is to apply a resource tag to them and to deploy to that resource tag first. For example, the scenario in the article Deploying applications with canary nodes applies the resource tag `canary` to one

node and the resource tag `mainSystem` to the other nodes. The article shows several ways to use those tags, but a simple way is to use a For Each Tag loop. This loop contains a simple sub-process that deploys the components and runs automated tests:


Start
Both canary and mainSystem tags
(For Each Tag)
Start
Install simpleComponent
Run automated tests
Run "Run automated tests"
Rollback simpleComponent
Finish

In the properties of the loop, you select the tags and then set the order for those tags. In this case, the `canary` tag runs first and the `mainSystem` tag runs second:



Generic processes

Generic processes are processes that are designed to run outside normal component or application processing. Generic processes are not specific to a particular component and can be used in any number of applications. They are like general purpose processes. You can use generic processes to download and install packages on the system or perform other configuration management related tasks such as configuring SSH or completing FTP tasks. With a generic process, for example, you can update Linux agents that were recently installed by using SSH to run as a service on a Linux computer. Linux agents cannot be updated in this way during a remote installation.

Generic processes can be used anywhere an application or component process might not be needed. Generic processes are run by agents on hosts that are managed by the agents. You can use generic processes in these instances, for example:

- In conjunction with a number of different component processes that do not fit well into a component template
- When you want to run processes that directly affect HCL DevOps Deploy (Deploy) agents that are unrelated to component, such as changing the agent configuration

Like component processes, generic processes consist of plug-in steps. Generic processes are created in the same way that component processes are created: steps are placed and configured in the process design editor. All plug-ins can be used, but some might be of little or no use outside a deployment. See Processes on page 2398.

To run a generic process, click **Processes**, and then click the generic process. On the **Dashboard** tab for the process, specify a resource (in most cases, an agent) in the **Resource** list, and then click **Submit**.

Generic processes can also be run from within component processes.

Starting in Deploy 7.0.3 generic processes can run against multiple resources at the same time. These resources can be selected at run time or in the process designer as part of the process flow.

Tip: You can set a default resource for generic processes:

1. Click the generic process to open it.
2. Open the **Configuration** tab.
3. Click **Basic Settings**.
4. In the **Default Resource** field, specify the default resource to run the process on. In most cases, specify an agent.
5. Click **Save**.

Importing and exporting generic processes

You can move generic processes between servers by exporting and importing them.

About this task

You can export a generic process from the **Processes** page to a JSON or YAML file containing the process, steps, and properties associated with the steps.

When you export components, applications, processes, or templates, and then import them on a different computer, the same versions of the same plug-ins must be installed on both computers.

You can export a generic process and import it into the another server.

1. On the first server, click **Processes** to open the **Processes** tab.
2. Export the process:
    a. Click the menu option of the process to be exported.
    b. Select **Export to JSON** or **Export to YAML**.

   You can load the file into a text editor or save it.

   Important: You must create the JSON file that contains the process definition from a server that uses the same version of HCL DevOps Deploy (Deploy) as the destination server.

   A file is created with the same name as the selected process.

   Example

   The file name takes the form of this example: `exampleprocess.json`.
3. Make the file available to the second server.
4. On the second server, click **Processes > Import Process** to open the **Import Process** window.
5. To upgrade an existing process, select **Upgrade Process**.

If the name of the process in the JSON file matches an existing process, the process parameters are updated with new values. If the process name is not found, the command has no effect.

Note: An existing process must match the name of the process in the JSON file, not the name of the file itself. The process name is the first parameter in the JSON file. The following text is the JSON representation of a process named `Post-process test`.

```
"name": "Post-process test",
```

6. Select an upgrade type from the dropdown list in the **Generic Process Upgrade Type for Linked Processes** field.
   - **Use Existing Process** type does not modify existing child processes that already exist on the system, but it still creates any child processes that are missing from the system but included in the import.
   - **Create Process** type generates a new process and appends a suffix to its name even if a process with the same name already exists during the import.
   - **Fail If Process Exists** type causes the import to fail if one of the processes being imported already exists on the system.
   - **Fail If Process Does Not Exist** type causes the import to fail if the process being imported does not already exist on the system.
   - **Upgrade Process if Exists** type updates any processes referenced by name to match the content of the import document and creates any processes that are missing.
7. Click **Choose File**, and then select the JSON file that contains the process definition.
8. Click **Submit**.

Results

You have exported and imported the generic processes between the servers.

Editing processes

In the process editor you organize the steps in a process, specify their properties, and connect them to each other.

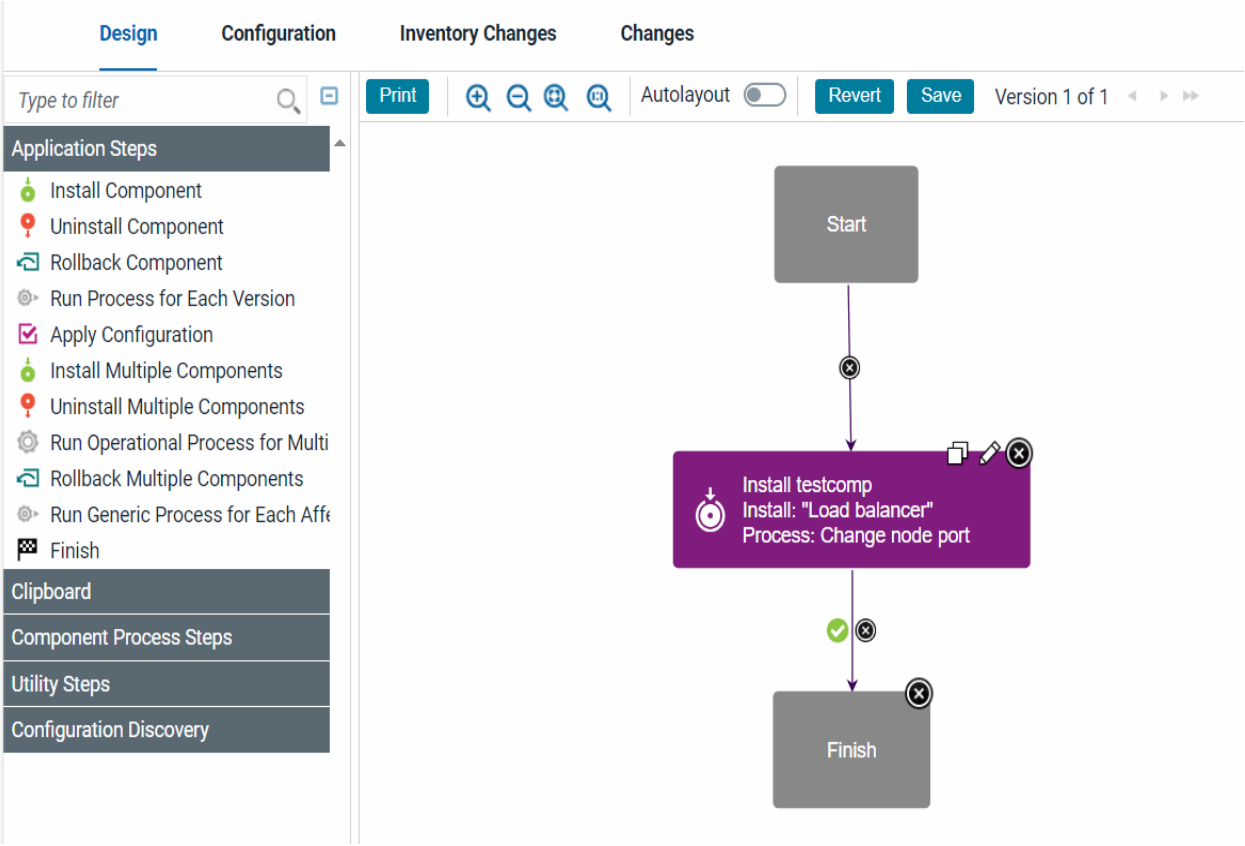
1. Open the component process in the process editor.

   Choose from:

   - To open a component process, click **Components**, click the component, click the **Processes** tab for the component, and click the process.
   - To open an application process, click **Applications**, click the **Processes** tab for the application, and click the process.
   - To open a generic process, click **Processes**, and click the process.

   Result

   The process editor opens. The editor is similar for each type of process, but the list of available steps is different for each type of process.



When the **Process Design** pane opens, the **Design** view is displayed. Processes are assembled with the **Design** view. Available steps are listed at the left of the window. Deploy provides several utility steps and plug-ins. More plug-ins and documentation for those plug-ins are available on: HCL plug-ins website.

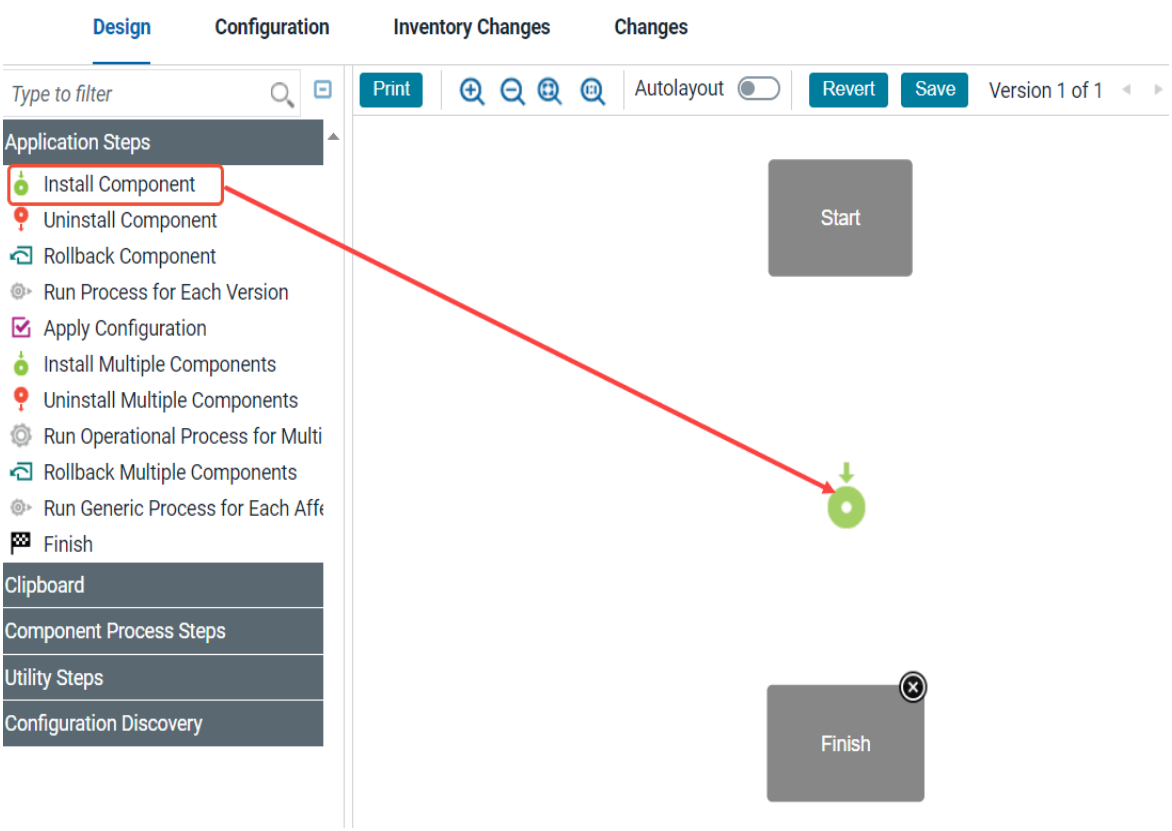
In addition to the process editor, the following two pages provide information about the process:

- The **Configuration** page shows the properties and settings for the process.
- The **Changes** page shows a record of every change to the process, addition or deletion of a property, and save or deletion of the process.

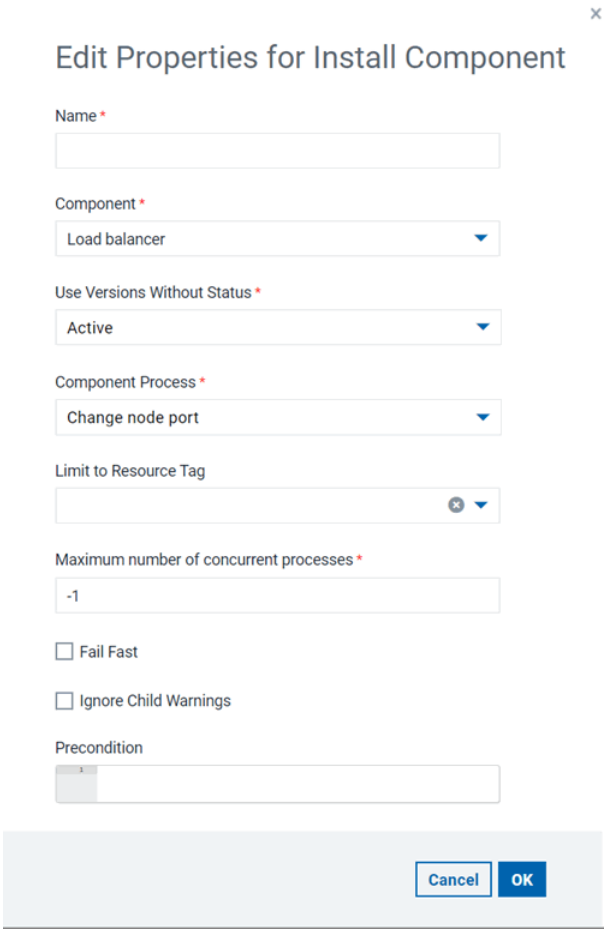
2. To add a step:

   a. From the list of steps at the left of the window, drag the step to use onto the design space.

   Result



To open the **Edit Properties** window, click the edit properties icon . When you drag the first property on the design pane, you can place the step anywhere on the design space. Connection links are automatically created from the Start node to the step and from the step to the Finish node. When you add more steps, drag them and position them so that they highlight the connection link between stesps so that the step is inserted and connection links are automatically created between the steps. Steps can be dragged and positioned at any time. However, you must delete the connection links from the step by clicking the delete connection icon , and then reconnect the moved steps.



Configuration dialogs boxes are tailored to the selected step. Only parameters that are associated with the step type are displayed.

b. Specify the properties for the step.
Each step has different properties. See Plug-ins on page 2556 for information about configuring specific steps.

c. **Optional:** To add or change the post-processing script, select a script in the **Post Processing Script** list. For more information, see Post-processing scripts on page 2415.

d. After you configure the step properties, save the step by clicking **Save**.

Result

The step is in the design space and is connected to other steps. If you change your mind, click **Cancel** to remove the step from the design space.

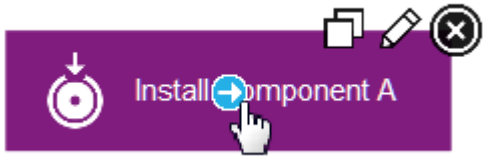
3. Verify that the step connections create the required process flow:
Connections control the process flow. The originating step runs before the target step. If you move steps or do not highlight a connection link when you place a step, creating a connection between steps is a simple process. You drag a connection from the originating step to the target step. Connections are typically formed automatically. However

when you create them manually, connections are formed one at a time between two steps, the originating step and the target step. To create connections manually:

a. Hover the cursor over the step to use as the origin of the connection.

Result

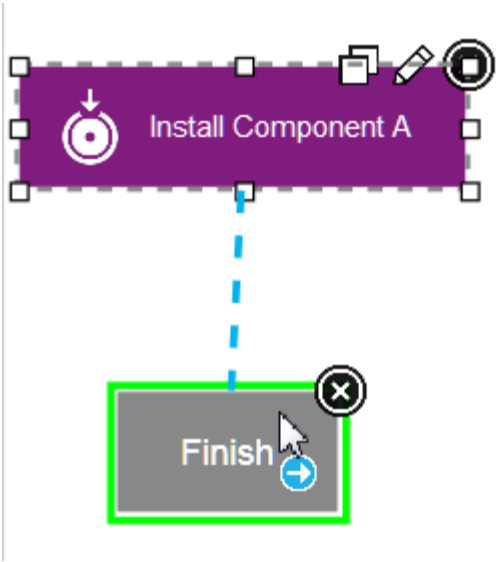
The connection tool is displayed.



b. Drag the connection tool over the target step.

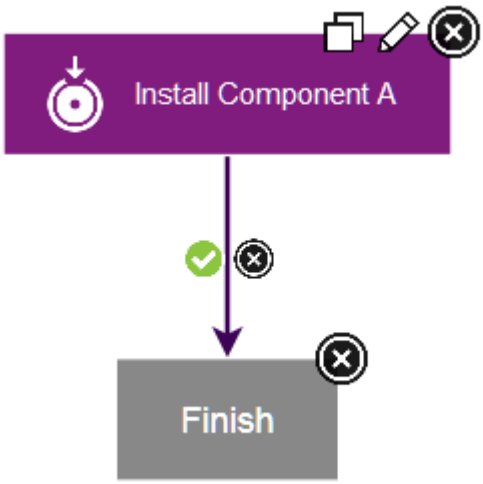
Result

The step beneath the connection tool is highlighted.



c. Release the connection tool over the target step to complete the connection.

Result



Each connection has a connection delete tool, *conditional flag*, and might have other tools depending on the originating step. Remove a connection by clicking the delete tool.

4. Add more steps to the process, connect them, and arrange them in the process editor.
   You can drag steps to move them around the process. In this way, you can arrange the steps in a meaningful arrangement on the process editor canvas.

   Tip: To move more than one step at a time, hold the Ctrl or Command key and click two or more steps to select them. Then, drag the steps to a different place in the editor.

5. Verify that the flow of steps is correct.
   Each process starts with the Start step and continues until it reaches the Finish step. Therefore, the Start step must connect to the first step or steps to run. Similarly, the last steps must connect to the Finish step, or else the process does not finish correctly.
6. When you are finished with your work on the process, click **Save**.
   If comments are required, the **Process Change Comment** window opens. Type a comment, and then click **Save**. The server settings control whether comments are required. To learn more about system settings, see Server settings and configuration on page 2069.

Working on processes in safe edit mode

Safe editing enables you to test component processes before making them available in normal environments. Make updates to your component process, test those updates safely, and then promote to a new version when you are ready.

Before you begin

Enable Safe Edit mode by completing the following steps:

1. Click **Settings > System Settings**. Select **Enable Safe Edit of Component Processes**, optionally select **Require Approval to Promote Component Process** and then click **Save** at the bottom of the System Settings page.
2. If **Require Approval to Promote Component Process** is enabled, after a draft is promoted you see **Promotion Pending** where the **Promote** push button was. The **Promotion Pending** button is inactive while an approval is pending. Approval requests go to anyone with manage processes permission on component. Under **Work Items**, there is a new Component Process promotion table with **Approve** and **Reject** actions. These actions link to latest promoted version and draft processes for which promotion was requested.
3. Two new parameters are displayed under the **Component Settings**.
    a. **Allow Process Locking**. When this option is enabled you will be able to lock component processes for editing.
    b. **Require Process Locking**. When this option is enabled you will not be able to edit component processes without locking the process first.

About this task

You can edit component processes and component template processes without affecting deployments to protected environments.

1. Under **Environments**, when you are editing the **Enable Drafts Processes** option, a new setting becomes available. After you select **Enable Draft Procees**, a new field, **Component using Process Drafts**, is displayed. Choose components for draft processes to run on.
2. On the **Components** tab, click the component name.
3. To **Edit** draft process, on the **Processes** tab, hover the cursor over the process to edit, and click the the pencil icon.
4. Design your draft process, and then click **Save**.
5. Test the process in the environment you set up in step one.
6. When process works as expected, go back to the process designer, and click **Promote**.
   This action makes the process available for use or creates a promotion approval.
7. **Optional:** Click **Lock** to lock draft.
   Result
   The name of the person who locked the draft is displayed under **Locked By** on the **Processes** page. Only the person who holds the lock can edit the process. Under **Actions** a **Force Unlock** option is available.
8. **Optional:** Click **Copy Latest Promoted Version To Draft** to create a copy of that version at the head of the draft stream. Click **Copy as Latest Draft** on a promoted process to create a copy of that version at the head of the draft stream.

Results

Environments that are set to allow draft processes can use draft process for deployments. Other environments must use the promoted processes.

Note: Snapshots can only lock to versions of component processes. They cannot lock to drafts. If a snapshot is deployed to an environment that allows drafts then one of these conditions apply:

- If the snapshot is using the latest version of a process, the draft will be used.
- If the snapshot is using a specific version of a process, then that specific version will be used.

Running processes

The way you run a process is different for each type of process.

About this task

To run a process, you must have permissions to access the processes and to run processes on the target agent resource.

Component processes

You cannot run a component process directly. Instead, you add one or more component processes to an application process and then run the application process.

Application processes

To run an application process, you must have an application environment. Click **Applications**, click the application, and then next to the environment, click **Request Process**. Then, specify the application process and the versions of the components to use. For more information, see Running application processes on page 2431.

Generic processes

To run a generic process, click **Processes**, and then click the generic process. On the **Dashboard** tab for the process, specify a resource (in most cases, an agent) in the **Resource** list, and then click **Submit**.

Tip: You can set a default resource for generic processes:

1. Click the generic process to open it.
2. Open the **Configuration** tab.
3. Click **Basic Settings**.
4. In the **Default Resource** field, specify the default resource to run the process on. In most cases, specify an agent.
5. Click **Save**.

Each time you request a process, the server keeps a log of information about the process and the individual logs from each step. To download these logs, open the process request and click **Download All Logs**.

Adding process properties

A *process property* provides a way to add user-supplied information to a process. A running process can prompt users for information and then incorporate it into the process. For example, if the process entails authenticating with another server, you can require that a user enter an ID and a password for the process to continue. Properties are defined in the **Edit Property** dialog box.

1. Open the process.
2. On the **Configuration** tab, click **Component Process Properties**, **Application Process Properties**, or **Process Properties**, depending on the type of process.
3. In the **Edit Properties** dialog box, enter a name in the **Name** field.
4. Optionally, enter a description in the **Description** field.
5. Enter a label in the **Label** field.
   The label is associated with the property in the user interface.

   Note: If two properties have the same label, they will not be grouped if the property names are not also the same.

6. In **Pattern**, enter a regular expression to require that user input values for this property match a specific pattern. Leave this field blank to accept any values.
7. If the property is required, check the **Required** check box.
   The default value is that the property is not required.
8. Specify the type of expected value with the **Type** list.
   Supported types are: `text`, `text area`, `check box`, `select`, `multi select`, and `secure`. Default type is `text`.
9. In the **Default Value** field, enter a default value (if any).
10. To save your work, click **Save**. To discard changes, click **Cancel**.

What to do next

To use a property in a process, reference it when you configure (see Component processes on page 2299) a step that uses it.

Copying process steps

You can copy and paste process steps in all process types: component, application, approval, and generic.

About this task

Steps that you copy are available until you delete them from the clipboard. Steps are saved to the clipboard on a per-user basis. Steps that you copy are not available in others' clipboards.

1. Open a process.
2. On the process step to copy, click the **copy** icon.
   The copy icon is to the left of the **edit** icon for the step: . Copying is not supported for the **For Each Agent** step in application processes.
   Result
   The process step is copied to the **Clipboard** folder in the **Step Palette**.
3. Expand the **Clipboard** folder in the **Step Palette**.
   Result

The steps that you copied are displayed. Only steps that apply to the current process are displayed in the **Clipboard** folder. For example, if you are editing an approval process, only the approval process steps that you copied to the clipboard are displayed.

4. To paste a step into a process, drag the step from the **Clipboard** folder to the design space.
   Click the **delete** icon on any step to remove the step from the **Clipboard** folder.

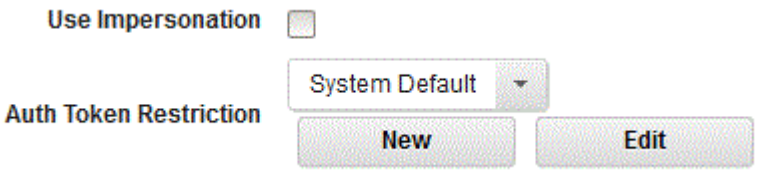
Manage process step authentication

Use token restrictions to manage process step authentication.

The HCL DevOps Deploy (Deploy) server generates an authentication token whenever it runs a plug-in process step. The token authenticates the user that is running the step and has the same permissions as that user. You can limit the actions available to tokens by applying token restrictions to them. A token restriction is a set of one or more REST API commands. When a token-restricted plug-in step runs, it uses the commands that are defined by the token restriction regardless of the user's permissions.

You apply token restrictions to plug-in steps in processes. You can apply a different token restriction to every plug-in step in a process. If you do not apply a restriction to a step, the default token restriction is used. The default token enables all the supported methods for all REST resources. You can modify the default token restriction but you cannot delete it.

You can apply token restrictions to component, application, and generic processes. To apply a token restriction to a process step, on the step configuration window, select the restriction from the **Auth Token Restriction** list.



You can edit the selected token restriction and also create a new token restriction from the plug-in step configuration window. For more information, see, Restricting authentication tokens on page 2223

User impersonation for process steps

HCL DevOps Deploy (Deploy) can use *user impersonation* when an agent must run a command for which it might not otherwise have permission, or when a specific user must be employed for a process.

On UNIX™ and Linux™ systems, the su and sudo commands are used to impersonate users; on Windows™ Deploy provides a utility program to handle impersonation. You implement impersonation when you configure a component's plug-in process step or resource.

Note:

When using impersonation, the user running the agent needs to have read, write, and execute access to the `<DEPLOY_AGENT_HOME>/var/work` directory.

The user starting the plugin as impersonating user needs to also have read, write, and execute access to the *<DEPLOY_AGENT_HOME>*`/var/work` directory.

Impersonation on Linux™ and UNIX™ systems

The su command (as used by HCL DevOps Deploy (Deploy)) enables a user to start a shell as another user (process steps can be considered individual shells).

When you configure a process step (see Processes on page 2398), you can tell Deploy to use impersonation for the step. By default, `su` is used but you can use sudo instead. To configure impersonation, you supply the user name that is required by the target host. When the impersonation-configured process step runs, the su or sudo command runs the step as the impersonated user. Each step that needs user impersonation must be configured independently.

Note: The user account must have root privileges in order to issue the su or sudo command when installing.

Before `sudo` can be used, impersonation privileges must be defined in the `/etc/sudoers` file. When you configure `sudoers`, ensure that the impersonating user does not need to supply a password. Typically, you would configure the `/etc/sudoers` file like the following example:

```
Defaults:X !requiretty
```

```
X ALL=(Y) NOPASSWD: ALL
```

where X and Y are user names. Configured this way, user X can run any command as user Y without supplying a password.

Also, Y can be a comma separated list, as shown here:

```
X ALL=(Y1, Y2, Y3) NOPASSWD: ALL
```

Note: The impersonated user must have read and execute access to the agent working directory.

As an alternative to configuring `sudoers`, in some cases you can configure the PAM via the module `pam_wheel.so` to allow certain users to log in without specifying a password. In this case, add users to a specific `wheel` group and edit the PAM configuration allow users in this group to run commands as though they were the root user.

su and sudo maintain a record in the system logs of all of their activity. su can be used without configuring the `sudoers` file. For information about su/sudo, see the UNIX™ or Linux™ documentation.

Note: For UNIX™ or Linux™ agents, the password option is ignored.

To customize the su and sudo commands, you can set the following properties on the agent, in the `installed.properties` file:

Table 732. Agent properties for the su and sudo commands

Property	Default value	Description
com.urbancode.shell.impersonation.unix.sudoFormat	`%s -n -u %u %c`	Syntax of the sudo command
com.urbancode.shell.impersonation.unix.sudoGroupFormat	`%s -n -u %u -g %g %c`	Syntax of the sudo command when a group is specified
com.urbancode.shell.impersonation.unix.suFormat	`%s - %u -c %c`	Syntax of the su command

Tip: To set these properties on new agents, create a generic process that updates the `installed.properties` file and run the process on new agents.

1. Create a generic process that appends the property to the file located at *<DEPLOY_AGENT_HOME>*`/conf/agent/installed.properties` (for example, use a Shell step to append the property to the file). Make sure you have the right file, the default working directory for a generic process is `\var\work`.
2. Once you've added the property, go to the **Resources -> Agents** tab, select the agents, and go to **Actions -> Restart** . This will restart the agent process and pick up the new property. It is recommended restarting no more than ~100 agents at a time to make sure they're all able to reconnect to the server quickly.

   If the agents are installed as services and you don't mind restarting the whole machine, you could alternatively use the Linux or Windows System tools plug-ins to restart the whole machine as part of the generic process, removing the need to restart all the agents from the UI.

   If the agents are started by a root user having system group, then the user running WAS plug-in should also be part of the system group or else the plug-in step fails with permission issue.
3. If later you need to run the generic process in bulk, you can create an application just for that purpose and add all the agents to an environment. You can then use the **For every agent** step to run a generic process on all the agents at once. It is recommended you should only do this process on a few agents so that you can test that everything is configured properly before moving to configuring the agents in bulk.

You can use the following variables in these properties:

Table 733. Variables for the su and sudo commands

Variable	Value
%s	The location of the su or sudo executable
%u	The user

Table 733. Variables for the su and sudo commands (continued)

Variable	Value
%g	The group
%c	The command to run

Impersonation on Windows™ systems

For agents that are running on Windows™, HCL DevOps Deploy (Deploy) provides a program that handles impersonation.

You implement impersonation for agents on Windows™ the same way you do for agents on UNIX™ or Linux™: when you configure a process step, you specify the user credentials that are used when the step is processed. These credentials are the user that the agent is impersonating. In this way, two user accounts are active: the user that the agent runs as normally and the user that the agent is impersonating while it runs the step.

To run steps on Windows™, the agent must be running as LocalSystem as a service or running in an administrator command prompt. Also, this user must have the following privileges, at a minimum:

- **SE_RESTORE_NAME** (Restore files and directories)
- **SE_BACKUP_NAME** (Back up files and directories)
- **SE_INCREASE_QUOTA_NAME** (Adjust memory quotas for a process)
- **SE_ASSIGNPRIMARYTOKEN_NAME** (Replace a process-level token)
- The user must also have access to the agent working directory, such as `<DEPLOY_AGENT_HOME>/var/work`.

Also, to impersonate an administrator, the agent user account must have the **SE_TCB_NAME** (Act as part of the operating system) permission.

The user that the agent is impersonating must have at least one of the following privileges:

- **SE_BATCH_LOGON_NAME** (Log on as a batch job)
- **SE_INTERACTIVE_LOGON_NAME** (Log on locally)
- **SE_SERVICE_LOGON_NAME** (Log on as a service)

In Windows™ Vista and later versions of Windows™, when the agent logs in as the impersonated user, the operating system returns a restricted token. If the impersonated user is a member of the Administrators group, the agent attempts to get an elevated token. However, if the agent user account does not have the **SE_TCB_NAME** permission, the agent does not receive an elevated token. In this case, the agent attempts to run the step with the restricted token, but it does not have administrator permissions.

You can also impersonate the Windows™ LocalSystem account. The LocalSystem account is installed on every Windows™ system and is the equivalent of the root user on UNIX™ and Linux™. It always has the privileges that are listed above.

Note: For agents on Windows™, the sudo option is ignored if selected.

Impersonation on z/OS® systems

The su command is used to impersonate users on agent computers that are running IBM® z/OS®.

The su command starts a shell as a different user. Process steps can be considered individual shells. When you configure a process step, you can specify that the step uses impersonation by selecting the **Use Impersonation** check box. To learn more, see Processes on page 2398. On z/OS® agent computers, the **Group**, **Password**, and **Use Sudo** fields that are displayed when you click **Use Impersonation** are ignored.

To configure impersonation, supply the user name that is required by the target computer. When the process step that is configured for impersonation runs, the su command runs the step as the impersonated user. You can configure user impersonation independently for each process step. You can also define user impersonation credentials for a resource. To learn more, see Defining default user impersonation credentials on page 2332.

You must configure a Resource Access Control Facility (RACF®) profile so that the su command can start a shell as a different user without specifying a password. The following RACF® profile grants authority to user X to impersonate user Y without entering a password:

```
RDEFINE SURROGAT BPX.SRV.Y UACC(NONE)
PERMIT BPX.SRV.Y CLASS(SURROGAT) ID(X) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

For more information about the su command, see the *z/OS® UNIX™ Systems Services* help.

Process steps: Reference

Application processes, component processes, and generic processes are created with the process editor.

HCL DevOps Deploy (Deploy) provides several common process steps. Otherwise, application processes are assembled from processes that are defined for their associated components. Application processes, like component processes, are created with the process editor.

The following process steps are provided by the server. For information about steps that are provided by plug-ins, see the documentation for the individual plug-in in the HCL site: https://hcl-software.com/plugins/

Acquire Lock

This step acquires a lock on a specified string value. You can use locks to prevent concurrent modification of resources.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

Upon reaching this step, the process searches for the lock with the specified name and proceeds as follows:

- Evaluates the lock status: If the lock already exists and the **Fail if Lock Exists** option is selected, the step fails.
- Creates the lock: If the step does not fail and the lock does not already exist, the process creates the lock.
- Acquires the lock: If no other process is using the lock, the step continues. If the lock is currently in use, the process waits until the lock is released before proceeding.

To release the lock, use the Release Lock on page 2483 step. Also, locks are released automatically when the process finishes. If multiple processes are waiting for the same lock, when the lock is released, the server selects the process that made the earliest request to acquire the lock.

Note: You can see the active locks by clicking **Settings > Locks**.

Table 734. Acquire Lock properties

Field	Description
Lock Name	A string value that identifies the lock. **Note:** Lock Name character limit is 4000.
Fail if Lock Exists	Select this checkbox to fail the step immediately if a lock with the specified name has already been created.

Add Inventory Status

With this process step, the component resource that the process is running against will have an inventory entry created for it with the specified status.

Tip:

Applications and components share some process steps. This step applies only to component processes.

Table 735. Add Inventory Status properties

Field	Description
Status	The component resource that the process is running against will have an inventory entry created for it with the specified status. `Active` indicates that the component version is

Table 735. Add Inventory Status properties (continued)

Field	Description
	deployed to its target resource. You can create other statuses and have those statuses applied to component versions when processes are complete. The status is displayed on the Inventory panes for the component itself and environments that ran the process.

Apply Configuration

This step runs a component process of the type Operational (No Version Needed) or Configuration Deployment.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes.

This step runs the specified component process if there is a difference in the configuration inventory of the resource and the configuration in the component process.

Table 736. Apply Configuration properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Maximum number of concurrent jobs	The maximum number component instances to apply the new configuration to simultaneously. For example, if you set the maximum number of concurrent processes to 2, the server applies the configuration to only two instances of a component at a time, even if many instances of the component are installed. To apply the configuration to every component instance simultaneously, specify `-1`. To limit the number of components to change at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to

Table 736. Apply Configuration properties (continued)

Field	Description
	check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Apply to Live Configuration

This step is a wrapper step for applying configuration updates to an application server.

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step does not apply to application processes that are associated with application templates. This step runs on the server, gathers information about the specified resource, and then calls a specified plug-in step on an agent computer. To use this step, you must install a supported application server plug-in, such as the WebSphere® Application Server - Configure plug-in. For information about plug-ins, see HCL™ Plug-ins.

Table 737. Apply to Live Configuration properties

Property	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Resource	The resource that contains the configuration information to apply to the application server.
Plugin Step	The plug-in step to run to apply the configuration to the application server. If no plug-ins are installed that support the Apply to Live Configuration process step, this list is empty.

Compare with Live Configuration

This step is a wrapper step for comparing the configuration of an HCL DevOps Deploy (Deploy) resource to a live application server.

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step does not apply to application processes that are associated with application templates. This step runs on the server, gathers information about the specified resource, and then calls a specified plug-in step on an agent computer. To use this step, you must install a supported application server plug-in, such as the WebSphere® Application Server - Configure plug-in. For information about plug-ins, see HCL™ Plug-ins.

Table 738. Compare with Live Configuration properties

Property	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Resource	The resource that contains the configuration information to compare with the application server.
Plugin Step	The plug-in step to run to compare the configuration to the application server. If no plug-ins are installed that support the Compare with Live Configuration process step, this list is empty.

Finish

Ends processing. A process can have more than one Finish step. All processes must reach a Finish step to complete successfully.

For Each Agent

This step creates a loop that iterates over the agents in the environment.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

This step provides vertical iterations, or iteration containers, for application processes. If, for example, you have numerous application servers, you can use this step to employ all agents in the environment to simultaneously deploy components to the servers. In this case, using this step means that overall deployment is not impacted even if one application server is slow.

This step runs the process that is defined within the container for each agent in the application environment. The workflow in the container has its own context, and its own start and finish steps, as shown in the following figure. You can use any application process step in the subprocess. An iteration container can connect to any step, even other iteration containers, but you cannot include other iteration containers within an iteration container.

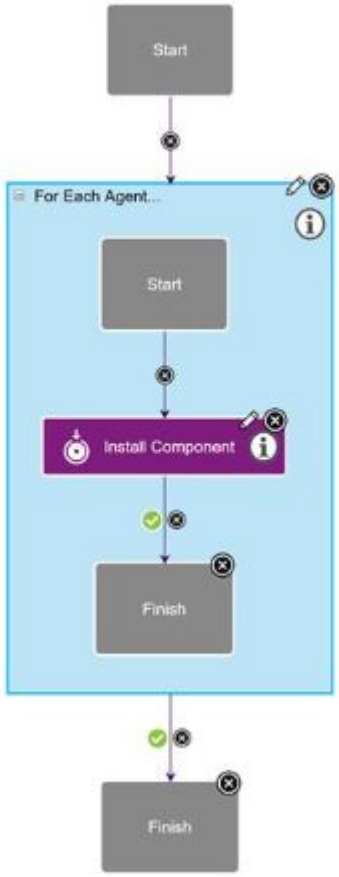
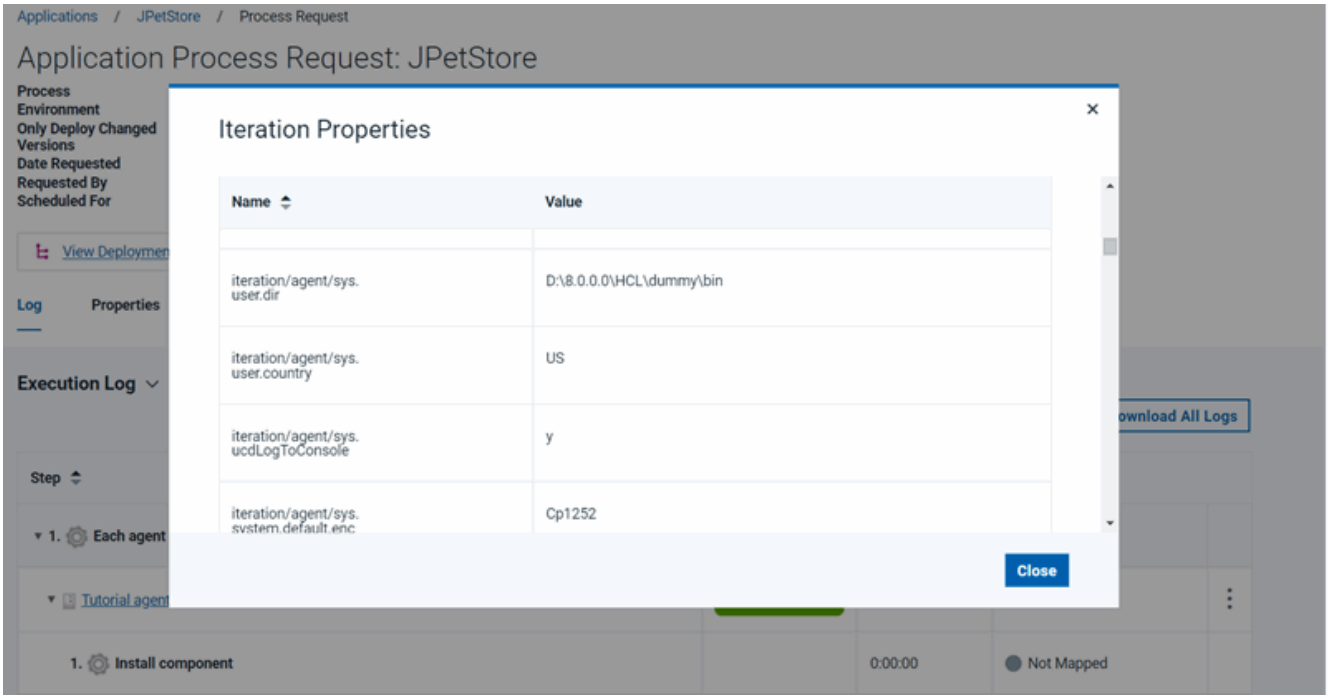


Table 739. For Each Agent properties

Field	Description
Name	A name for the iteration container.
Limit to Resource Tags	A list of resource tags that limit the container. The subprocess runs only on the agent resources that have one or more of the specified resource tags.
Maximum number of concurrent processes	The maximum number of agents that run the subprocess at a time. To run the subprocess on every agent at once, specify `-1`. To limit the number of agents to deploy to at once, as in blue/green deployments or rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Iteration properties

The For Each Agent step has properties, called iteration properties, that can be used to reference steps within the iteration context. By using these properties, for example, you can identify the current agent within the context. To reference iteration properties, use the following format: `${p:iteration/<name>}`.



To view the iteration properties for the step, follow these steps:

1. Open the log for the application process request by clicking **Applications > application_name > History > View Request**.
2. Expand the container step and, next to one of the iterations, click **Iteration Properties**.

For Each Tag

This step iterates over resource tags in application processes.

Tip:

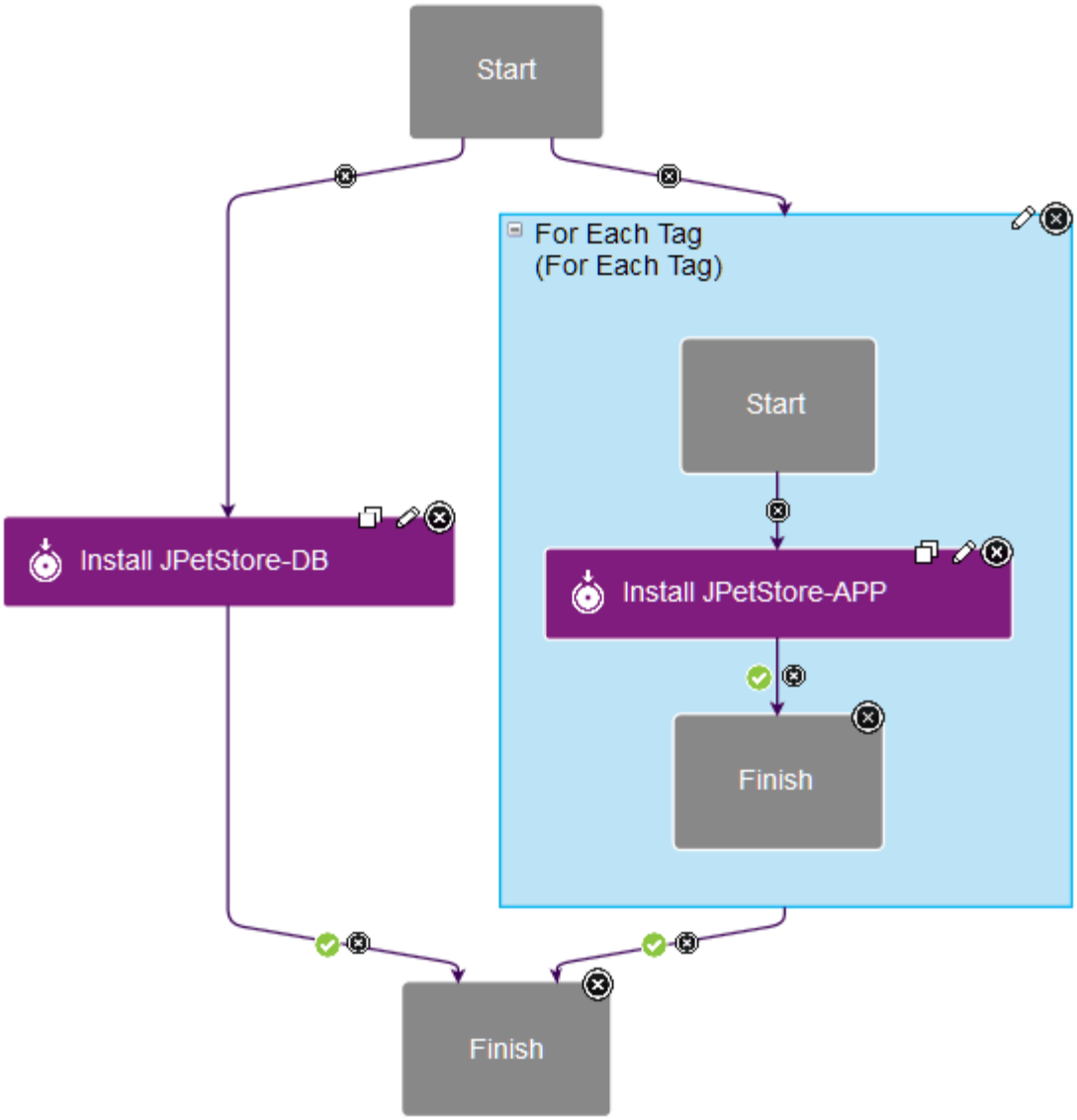
Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

This step iterates over one or more resource tags. If you have many agent resources mapped to an environment, you can use this step to choose which resources to run steps on. For example, if you are running a rolling deployment, you can use this step to deploy to a few resources first, and then run automated tests on those resources before deploying on the other resources. Also, if you are running a blue/green deployment, you can use this step to limit the deployment to the blue nodes

or green nodes. Finally, you can select the order of the tags to iterate over, such as if you want to update specific canary nodes before all other nodes. For a tutorial of Blue/Green deployments, see Blue/Green deployments.

Note: If the process fails in a For each tag step for any agent or agent pool, no new agents will have the process run for them and the step will fail.

Like the For Each Agent step, this step creates an iteration container for subprocesses. You specify one or more resource tags, and the order of those resource tags, and the step runs the subprocess once for each resource tag. The workflow in the container has its own context, and its own start and finish steps, as shown in the following figure. You can use any application process step in the subprocess. An iteration container can connect to any step, even other iteration containers, but you cannot include other iteration containers within an iteration container.



To make it easier to control which resources you deploy to, such as in a staged, rolling, or blue/green deployment scenario, you can set the tags on the fly when you request the process. In the **Request Process** window, the list of tags for the container is shown. You can add or remove tags here, or change the order of the tags. This way, you can use one process to deploy in different scenarios. For example, if you want to deploy to a canary node first, put that resource tag first, and then follow with the other tags. The following figure shows a process request to a canary node first, and to the rest of the resources next.

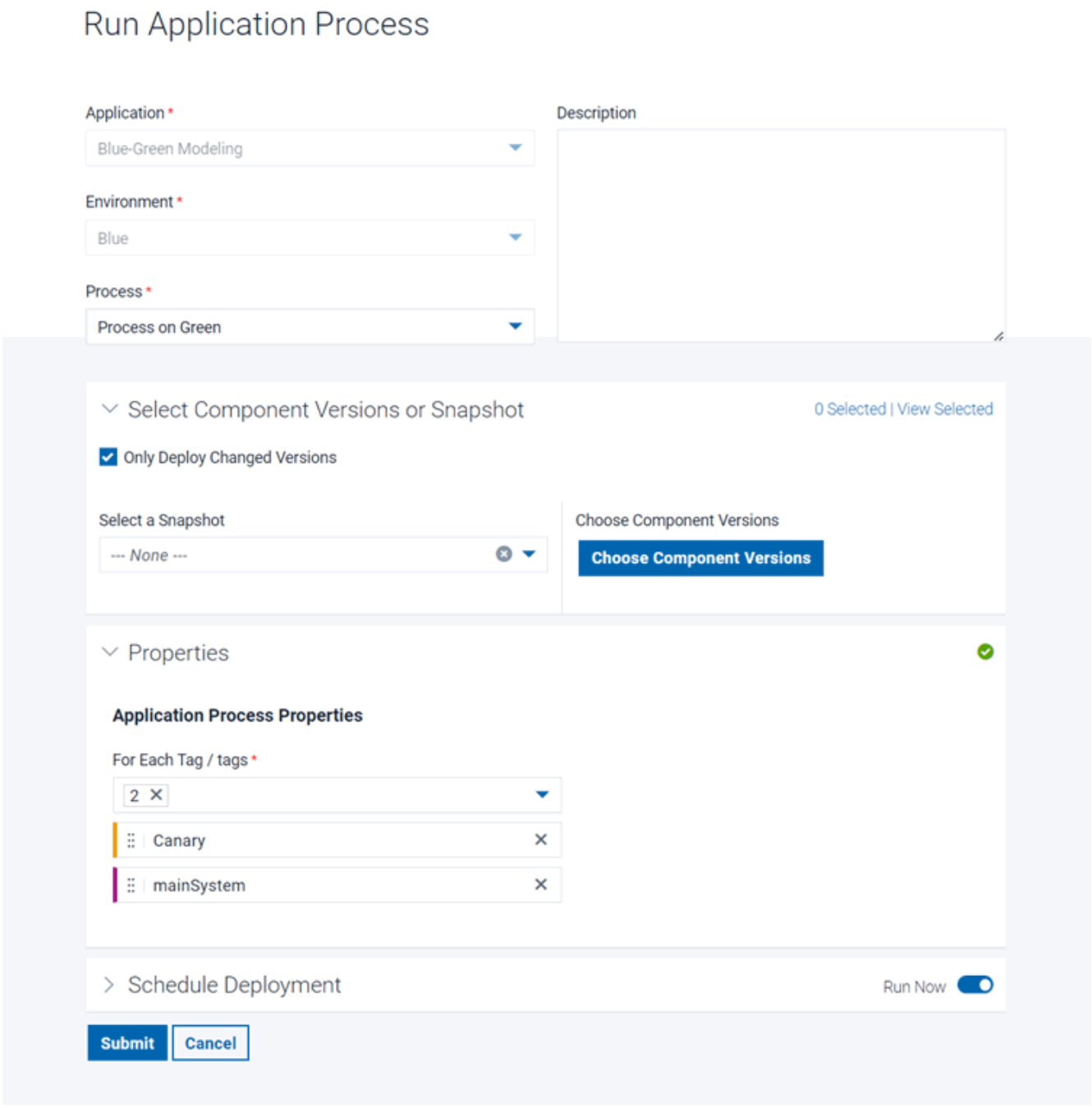


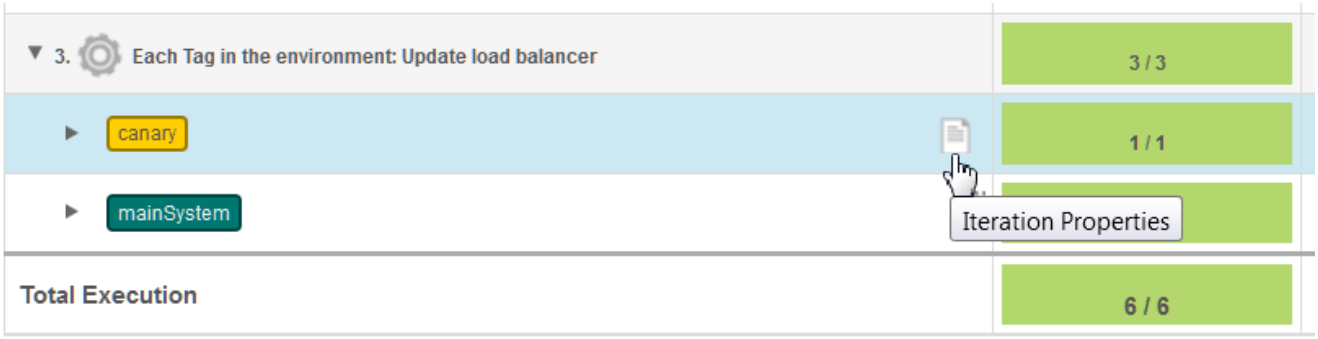
Table 740. For Each Tag properties

Field	Description
Name	A name for the iteration container.
Tags	A list of resource tags to use in the container. The subprocess runs once for each resource tag. You can order the resource tags by dragging and dropping them.
Max Concurrent Tags	The maximum number of tags for which to run the subprocess at a time. To run the subprocess on all of the tags at the same time, specify `-1`. To run the subprocess on only a few of the tags at a time, as in blue/green deployments or rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Iteration properties

The For Each Tag step has properties, called iteration properties, that can be used to reference steps within the iteration context. By using these properties, for example, you can identify the current agent within the context. To reference iteration properties, use the following format: `${p:iteration/<name>}`. For example, the current tag name is in the property `${p:iteration/tag.name}`.

When you run the application process, the log shows each iteration of the container as a separate entry. For example, the following log shows a For Each Tag step that ran over two tags:



To view the iteration properties for the step, follow these steps:

1. Open the log for the application process request by clicking **Applications > application_name > History > View Request**.
2. Expand the container step and, next to one of the iterations, click **Iteration Properties**.

Note: If the process fails in a For Each Tag step for any agent or agent pool, no new agents will have the process run for them and the step will fail.

Related information

Blue-green deployments on page 1984

Install Component

This step installs the selected component with one of the processes that are defined for the component.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes.

This step runs the specified component process for all selected component versions, except the component versions that are already in the inventory. By contrast, the Run Process for Each Version on page 2492 step runs the process for each component version, whether or not each component version is in the inventory.

It is up to you to select an appropriate component process to run. In most cases, select a component process of the Deployment type. If you select a different type of component process, you might get results that you do not expect. The server runs whatever component process you select, even if that process does not install the component. For example, if you select a component process of the Uninstall type, the Install Component step runs that component process as usual. Then, the component process uninstalls the component and removes the component from the inventory.

This step can install multiple incremental versions of the same component. To do this, use a single Install Component step in the application process. Then, when you run the application process, select multiple incremental versions to deploy. The incremental versions are deployed in the order that you select them in the **Run Process** window. To deploy the incremental versions in a specific order, you can also create a snapshot, add the component versions to the snapshot in the order that you want to deploy them, and run the application process with the snapshot.

Table 741. Install Component properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. To install another component, add another install step to the process.
Use Versions Without Status	Restricts the components that can be used by the step. Components with the selected status are ignored. Available statuses: `Active` means ignore components that are currently deployed.
Component Process	The component process to run.
Limit to Tag	User-defined roles can be used in processes; see Adding tags to objects on page 2521.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously to deploy the component. For example, if you set the maximum number of concurrent processes to 2, only two instances of the component are installed at a time, even if many instances of the component are mapped to agents. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Install Multiple Components

Install multiple components that are based on component tags or resource tags.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

It is up to you to select an appropriate component process to run. In most cases, select a component process of the Deployment type. If you select a different type of component process, you might get results that you do not expect. The server runs whatever component process you select, even if that process does not install the component. For example, if you select a component process of the Uninstall type, the Install Component step runs that component process as usual. Then, the component process uninstalls the component and removes the component from the inventory.

Table 742. Install Multiple Components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	A process for the components that contain the selected tag. The process name must be the same for each tagged component. Only components that contain the specified process run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Use Versions Without Status	Restricts the components that can be used by the step. Components with the selected status are ignored. Available statuses: `Active` means ignore components that are currently deployed.
Maximum number of concurrent	The maximum number of components to deploy at one time. For example, if you specify 5, this step deploys five components at a time to all agents to which the components are mapped. To run this step on all components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Table 742. Install Multiple Components properties (continued)

Field	Description
components	**Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to deploy each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are installed at a time, even if many instances of the component are mapped to agents. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Inventory Update

This step updates the inventory and compliance for the environment.

Restriction: The Component Inventory Update and Snapshot Inventory Update steps are available only in application processes that have **Inventory Management** set to **Advanced**. See Managing environment inventory and compliance on page 2396. This step is available only to processes that have the **Inventory Management** check box set to **Advanced**.

Table 743. Inventory Update properties

Field	Description
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available.
Limit to Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.

Table 743. Inventory Update properties (continued)

Field	Description
Action	Select **Add Desired Inventory** to add component versions to the inventory. Select **Remove Desired Inventory** to remove component versions from the inventory.

Join

The Join step merges a process so that only one step runs at a time. If any of the steps that connect to a Join step fail or do not run, the Join step causes the process to fail.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

For information about this step, see Branching and joining steps on page 2402.

This step has no properties.

Lock Snapshot

This step locks an application snapshot version to prevent its component configurations from being modified or overwritten.

When a snapshot is locked, its component versions become immutable. If the specified snapshot is already locked, the step completes successfully without making changes.

This step is available for use in application processes, component processes, and generic processes.

Table 744. Lock Snapshot properties

Field	Description
Snapshot Name or ID	The name or unique ID of the snapshot to lock. You can specify a name or use a property reference defined in the process, such as `${p:snapshot.name}`.
Application Name or ID	The name or unique ID of the application associated with the snapshot. You can specify a name or use a property reference defined in the process, such as `${p:application.id}`.

Related reference

Acquire Lock on page 2466

Related information

Creating snapshots on page 2383

Manual Application Task (Utility)

A manual task interrupts an application process until manual intervention is done. A task-interrupted process remains suspended until the targeted user or users respond. Typically, manual tasks are removed after the process is tested or automated.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates. However, the generic process step includes the **Comment Prompt** field.

Similar to approvals, triggered tasks alert targeted users. Alerts are associated with environment- or application-defined user roles. Affected users can respond (approve) by using the **Work Items** pane (see Responding to approval requests on page 2394). Unlike approvals, manual tasks can be incorporated within an application process.

For information on using manual tasks in application processes, see Manual application tasks on page 2387.

Table 745. Manual Application Task properties

Field	Description
Name	The name for the manual task.
Only Allow Deploying User To Complete	Restrict approval to the user who started the deployment.
Notification Template	Specify a notification template. The individual tasks map to the notification scheme used by the application (see Creating Notifications in a Notification Scheme on page 2081). If you do not want notifications created for the task, select **None**.
Properties	Optionally, add one or more properties to the task.
Require Comment	Select to require that users must add a comment about this manual task.
Comment Prompt	**Note:** This field applies only to generic process steps. Enter the text that requests users to add a comment about this manual task.

Note

The Note step adds a note to a process diagram.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

This step has a single property, which is the text of the note.

Operational component processes

Runs the specified operational (no version needed) component process.

You can add operational component processes to application processes if the operational component processes do not require a component version. Operational component processes that do not require versions are listed in the step palette under **Component Process Steps**. To add an operational (no version needed) component process to an application process, drag the folder that represents the component to the application process and then select the operation process to run. To run an operational (with version) component process, use the Run Process for Each Version on page 2492 step.

To learn about the different types of component processes, see Component process types on page 2423.

Table 746. Operational component process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Limit to Tag	Run the process only on the versions with the specified tag.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously. For example, if you set the maximum number of concurrent processes to 2, the process runs on only two instances of a component at a time, even if many instances of the component are installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Run on First Onl	Instead of being run by all agents that are mapped to the environment, the step runs only on the first online agent that the server identifies. The mechanism that is used to identify the "first" agent is database-dependent, and thus indeterminate. If no agents are online, the server skips the step and marks its status as "Not Mapped." This setting

Table 746. Operational component process properties (continued)

Field	Description
ine Reso urce Only	may cause the environment to become noncompliant or to provide unexpected results, because only one agent runs the process, even if many agents are mapped to the environment.
Preco ndit ion	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Run If Comp one nts Cha nge	When this check box is selected, this step runs only if an agent resource that is mapped to this component has other component resource mappings whose inventories are changed by the process. When you select this check box, you can select components and specify whether the step runs on all resources for the component that contains this process, or only on resources that include both the component for this process and one of the selected components that was changed by this process. **Resource Selection Mode** options: • To run the operational process only on resources that have had their inventory updated for one of the selected components, select **Only resources where selected components are changed**. For example, when you have a component that contains some operational processes, for example, "Restart HTTP Server", you can add the operational process to the application process and you can select this option to run only on machines where you have deployed changes to web sites being hosted. • To run the operational process wherever the component defining the process is added to any agents in the environment, select **All resources mapped to this process's component**. For example, if after you deploy changes to a web site you want to restart a load balancer that is on a different machine than the web site, you can select this option to run the operational process wherever the "Load Balancer" component that contains the "Restart Load Balancer" operational process is mapped.

Release Lock

This step releases a lock on a specified string value.

Tip: Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

To create a lock, use the Acquire Lock on page 2466 step. If multiple processes are waiting for the same lock, when the lock is released, the server selects the process that made the earliest request to acquire the lock.

Table 747. Release Lock properties

Field	Description
Lock Name	A string value that identifies the lock.

Remove Inventory Status

With this process step, the component resource that the process is running against will have an inventory entry removed from it.

Tip:

Applications and components share some process steps. This step applies only to component processes.

Table 748. Remove Inventory Status properties

Property	Description
Status	The component resource that the process is running against will have the specified inventory entry removed from it.

Rollback Component

Use this process step to roll back a component version to the version in a snapshot or to the version that was deployed when the process began.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes.

For a video demonstration of rolling back components, see Rollback Scenarios in Deploy.

In most cases, to switch to a different version of a component, you deploy the new version over the existing version. If necessary, you can run the uninstall process for the component first.

However, you can also use the **Rollback Component** step to automate the process of switching to a previous component version. This step has two options:

- **Remove Undesired Incremental Versions**: This option removes versions that are not in a specified snapshot. See Rolling back to a snapshot on page 2437.
- **Replace with Last Deployed**: This option rolls back components that fail to deploy. See Uninstalling components with an application process on page 2436.

Table 749. Rollback Component properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. If you want to roll back another component, add another rollback step to the process or use the Rollback Multiple Components on page 2486 step.
Remove Versions with Status	Restricts the component versions that are affected by the step. Only versions with the selected status are affected. By default, the status `Active` refers to versions that are currently deployed.
Component Process	The component process to run. In most cases, select a component process that uninstalls the component.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Rollback type	Determines the type of rollback. If you specify `Remove Undesired Incremental Versions`, the server runs the uninstall process for each version that is not in the specified snapshot. See Rolling back to a snapshot on page 2437. If you specify `Replace with Last Deployed`, the server runs the specified process on the version that was installed when the process began. See Uninstalling components with an application process on page 2436.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are rolled back at a time, even if there are many instances of the component. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Rollback Multiple Components

Use this step to roll back multiple component versions to versions in a snapshot or to versions that were deployed when the process began. The multiple components are based on component tags or resource tags.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

In most cases, to switch to different component versions, you deploy the new versions over the existing versions. If necessary, you can run the uninstall process for the components first.

However, you can also use the Rollback Component step to automate the process of switching to a previous component version. This step has two options:

- **Remove Undesired Incremental Versions**: This option removes versions that are not in a specified snapshot. See Rolling back to a snapshot on page 2437.
- **Replace with Last Deployed**: This option rolls back components that fail to deploy. See Uninstalling components with an application process on page 2436.

Table 750. Rollback Multiple Components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	The component process to run. In most cases, select a component process that uninstalls the component.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Remove Versions With Status	Restricts the components that are affected by the step. Only components with the selected status are affected. Available statuses: `Active` refers to components that are currently deployed.
Rollback Type	Determines the type of rollback. If you specify `Remove Undesired Incremental Versions`, the server runs the uninstall process for each version that is not in the specified snapshot. See Rolling back to a snapshot on page 2437. If you specify `Replace with Last Deployed`, the server runs the specified process on the version that was installed when the process began. See Uninstalling components with an application process on page 2436.

Table 750. Rollback Multiple Components properties (continued)

Field	Description
Maximum number of concurrent components	The maximum number of components to roll back at one time. For example, if you specify 5, this step rolls back five components at a time on all agents to which the components are installed. To run this step on all components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to roll back each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are rolled back at a time, even if there are many instances of the component. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Run Component Process

You can run a component process as a part of a step.

Tip: Applications and components share some process steps. This step applies only to component processes.

Important: If you have not created another component process before you attempt to place this step into the current process you receive this message: `This will call the current process and risk the creation of an infinite loop. Are you sure you want this process to run itself?` You can close the message window and work with the properties. However, the only process that you can select in **Component Process** is the current process.

Table 751.

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Process	The component process to run.

Run Generic Process

This step runs a generic process as a step in an application process.

Tip: Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

Table 752. Run Generic Process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Process	The generic process to run.
Resource Path	The resource on which to run the generic process.
Other properties	After you select a generic process, the properties for that process are listed. You can set values here to pass to the process.

Run Generic Process for Each Affected Resource

Runs the specified generic process on each resource that is affected by the application process.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

Table 753. Install Component properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Process	The generic process to run.
Maximum number of concurrent processes	The maximum number of concurrent processes to run at the same time. For example, if you set the maximum number of concurrent processes to 2, the generic process runs on only two resources at a time. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Other properties	After you select a generic process, the properties for that process are listed. You can set values here to pass to the process.

Run Operational Process for Multiple Components

Runs the operational (no version needed) process for multiple components.

You can add multiple operational component processes to application processes if the operational component processes do not require a component version. To learn about the different types of component processes, see Component process types on page 2423. To run a single operational process, see Operational component processes on page 2482.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

Table 754. Operational component process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	A process for the components that contain the selected tag. Only components that contain the process will run. Only one process can be selected per step.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Maximum number of concurrent comp	The maximum number of components on which to run processes at one time. For example, if you specify 5, this step runs processes on five components at a time. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Table 754. Operational component process properties (continued)

Field	Description
onents	
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously. For example, if you set the maximum number of concurrent processes to 2, the process runs on only two instances of a component at a time, even if many instances of the component are installed. To run an unlimited number of concurrent processes, specify `-1` . To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Run on First Online Resource Only	Instead of being run by all agents that are mapped to the environment, the step runs only on the first online agent that the server identifies. The mechanism that is used to identify the "first" agent is database-dependent, and thus indeterminate. If no agents are online, the server skips the step and marks its status as "Not Mapped." This setting may cause the environment to become noncompliant or to provide unexpected results, because only one agent runs the process, even if many agents are mapped to the environment.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Run If Comp	To run this step only if another step in the application process changes the inventory of a component, select this option. If you select a component tag, for this step to run, a component that contains that tag must have an inventory change. If you do not select a component tag, any inventory change that is caused by the application process causes this step to run. **Note:** Do not select the same component tag that you selected in the **Component Tag** list. Select

Table 754. Operational component process properties (continued)

Field	Description
on ent with Tag Cha n ges	an option in the **Resource Selection Mode** list to control the agent resources that this step runs on. To run the step on only resources where the inventory of other components changes, select **Only resources where component with selected tag is changed**. For example, you might have a component that contains some operational processes, such as "Restart HTTP Server." If you add the operational process to the application process, you can select this option to run only on machines where you are deploying changes to websites that are hosted. To run the operational process on all possible resources, select **All resources mapped to this process's component**. For example, if you deploy changes to a website, you might want to restart a load balancer that is on a different machine than the website. You can select this option to run the operational process wherever the "Load Balancer" component that contains the "Restart Load Balancer" operational process is mapped.

Run Process for Each Version

Use this step to run the specified process once for each version of the specified component.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes.

This step runs the specified component process for all selected versions, whether or not each component version is in the inventory. By contrast, the Install Component on page 2476 step runs the component process only for component versions that are not already in the inventory.

Table 755. Run Process for Each Version properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Compon ent	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available.
Compon ent Process	The component process to run.
Limit to Resou rce Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Maxi mum number	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously on each component. For example, if you set the maximum number of concurrent processes to 2, the server accesses only two instances of a component at a time, even if many

Table 755. Run Process for Each Version properties (continued)

Field	Description
of concurrent processes	instances of the component are installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Run on First Online Resource Only	Instead of being run by all agents that are mapped to the environment, the step runs only on the first online agent that the server identifies. The mechanism that is used to identify the "first" agent is database-dependent, and thus indeterminate. If no agents are online, the server skips the step and marks its status as "Not Mapped." This setting may cause the environment to become noncompliant or to provide unexpected results, because only one agent runs the process, even if many agents are mapped to the environment.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Set Status

This step sets the ending status of the process. The step can specify that the process is completed or failed.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

Table 756. Set Status property

Field	Description
Status	`Failure` or `Success`.

Switch

Use this step to create branches in the process that are based on the value of a property.

Tip:

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step applies to application processes that are associated with application templates.

For information about this step, see Switch steps on page 2400.

Table 757. Switch properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Property Name	The property to use for the branching. You can specify the property name by itself or in the same form as you would use in other steps, such as `${p:propertyName}`. You cannot use a secure property in this field.

Uninstall Component

Use this step to uninstall the selected component.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes.

It is up to you to select an appropriate component process to run. In most cases, select a component process of the Uninstall type. If you select a different type of component process, you might get results that you do not expect. The server runs whatever component process you select, even if that process does not uninstall the component. For example, if you select a component process of the Deployment type, the Uninstall Component step runs that component process as usual. Then, the component process deploys the component and adds the component to the inventory.

Table 758. Uninstall Component Properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. To uninstall more than one component, add another uninstall step to the process.
Remove Vers	Restricts the component versions that are affected by the step. Only versions with the selected status are affected. By default, the status `Active` refers to versions that are currently deployed.

Table 758. Uninstall Component Properties (continued)

Field	Description
ions with Status	
Component Process	The component process to run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Uninstall Type	The **All Existing** option removes all the versions of this component that are in the inventory. The **All Selected For Process** option removes only the versions that you specify when you start the application process.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously to uninstall component instances. For example, if you set the maximum number of concurrent processes to 2, only two instances of the component are uninstalled at a time, even if many instances of the component are currently installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Uninstall Multiple Components

Use this step to uninstall multiple components that are based on component tags or resource tags.

Tip:

Applications, components, and generic processes share some process steps. This step applies only to application processes, including application processes that are associated with application templates.

It is up to you to select an appropriate component process to run. In most cases, select a component process of the Uninstall type. If you select a different type of component process, you might get results that you do not expect. The server runs whatever component process you select, even if that process does not uninstall the component. For example, if you select a component process of the Deployment type, the Uninstall Component step runs that component process as usual. Then, the component process deploys the component and adds the component to the inventory.

Table 759. Uninstall Multiple Components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	The component process to run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Remove Vers	Restricts the components that are affected by the step. Only components with the selected status are affected. Available statuses: `Active` refers to components that are currently deployed.

Table 759. Uninstall Multiple Components properties (continued)

Field	Description
ions With Status	
Maximum number of concurrent components	The maximum number of components to uninstall at one time. For example, if you specify 5, this step runs the uninstall component processes for five components at a time. To run this step on all components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to uninstall each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are uninstalled at a time, even if many instances of the component are currently installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Uninstall	The **All Existing** option removes all the versions of this component that are in the inventory. The **All Selected For Process** option removes only the versions that you specify when you start the application process.

Table 759. Uninstall Multiple Components properties (continued)

Field	Description
Type	

Update from Live Configuration

This step is a wrapper step for updating a resource with the configuration information from an application server.

Applications, components, and generic processes share some process steps. This step applies to application processes, component processes, and generic processes. This step does not apply to application processes that are associated with application templates. This step runs on the server, gathers information about the specified resource, and then calls a specified plug-in step on an agent computer. To use this step, you must install a supported application server plug-in, such as the WebSphere® Application Server - Configure plug-in or the WebSphere® Application Server - Deploy plug-in. For information about plug-ins, see HCL™ Plug-ins.

Table 760. Update from Live Configuration properties

Property	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Resource	The resource to use to store the configuration information that is retrieved from the application server.
Plugin Step	The plug-in step to run to retrieve the configuration information from the application server. If no plug-ins are installed that support the Update from Live Configuration process step, this list is empty.

Statuses

Create and apply statuses to track component inventory states or to track component versions in environments and resources.

Statuses can be used to track component version and component inventory states. Version statuses are used with application gates (see Environment gates on page 2360) to ensure that only component versions that meet certain criteria are deployed. Inventory statuses can track component versions in environments and resources. You can create inventory statuses for any requirement.

You apply a version status to a component that you created. You apply an inventory status when you configure a component process. See Creating component processes on page 2425.

Creating statuses

Define component version, snapshot, and inventory statuses on the **Statuses** tab.

1. Click **Settings > Statuses**.
2. For the status type to create, click **Add Status**.
3. Configure the status in the **Add Status** window:

Parameter	Description
Name	Identifies the status and is included in many UI elements.
Description	Conveys more information about the process.
Color	Defines the status color that is displayed in the UI.
Unique	When checked, specifies that only a single instance of the status describes a component. For inventory or snapshot statuses, an application removes the status from existing versions in the environment or resource inventory. For version or snapshot statuses, the status can be used by only one version or snapshot at a time.
Required Role	Specifies the user role that is necessary for a user to assign the status.

4. Save your work when finished.
   Statuses are stored in the HCL DevOps Deploy (Deploy) database.

Tip: Default statuses are defined in an XML file, which you can edit to add your own values.

Adding version statuses

Manage component version statuses through the version options interface.

About this task

You add or remove statuses from individual component versions to provide information about the version. If you copy files to an agent without deploying them, you might apply a `Staged` status to them. When you install the version, you might set the status to `Active`.

1. To select the component, click **Components > *selected component***.
2. To select the version, click **Versions > *selected version***.
3. Click **Add a Status**.
4. Select the status from the list, and click **Save**.

What to do next

Add a status by using the Add Status to Version plug-in step, which is available from the Version plug-in.

Adding snapshot statuses

Manage snapshot statuses through the snapshot options interface.

About this task

You add a status to a snapshot to provide information about the snapshot deployment status. If a snapshot deployment fails, you might apply a `Failed` status to them. When a snapshot is deployed, you might set the status to `Active`.

1. Click **Applications > *selected application***.
2. To select the snapshot, click **Snapshots > *selected snapshot***.
3. Click **Add Status**.
   The **Add a status** window is displayed.
4. Select a status from the list, and click **Save**.

What to do next

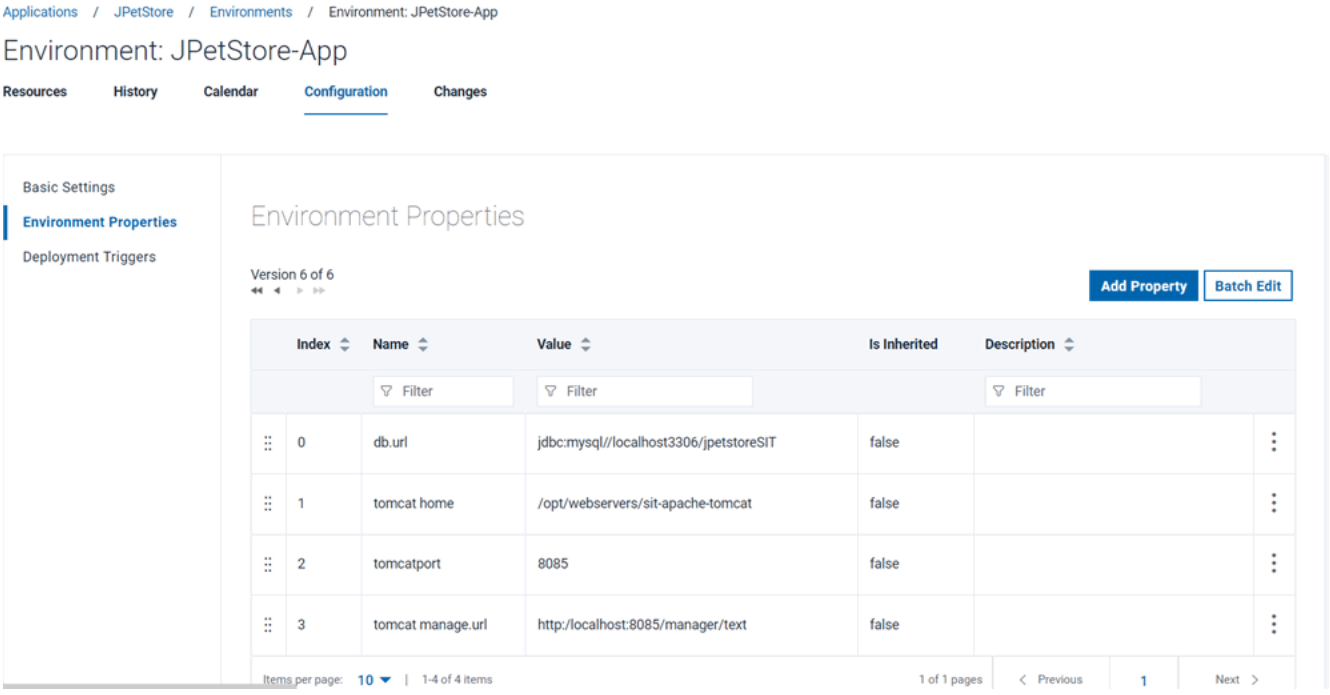
You can now create snapshot gates using the snapshot statuses.

Properties

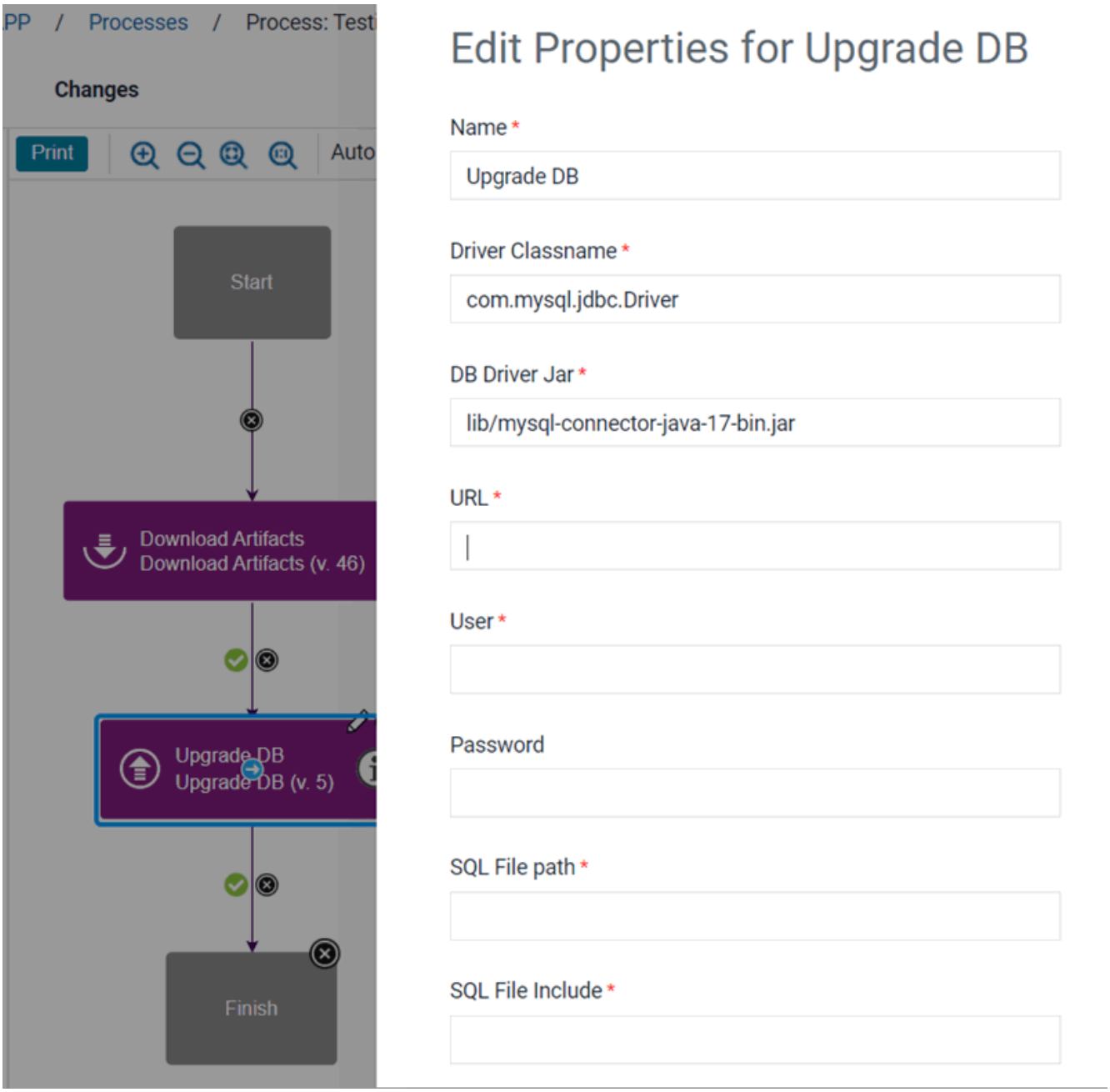
Properties are variables that store information about many different elements, including components, environments, processes, and applications. You can also set global properties for the system.

Common uses of properties are to store the locations of files, folders, servers, tools, or other resources that might change. For example, the application in the tutorial Deploying a simple web application on page 1934 uses properties to store the location of a web server and a database. In the tutorial, you set the location of a database in an environment property that is named `db.url`. Then, any time that you want to refer to the location of the database, such as when you configure an application to store data in that database, you can use the property value `${p:environment/db.url}`. If you need to change the location of the database, you only need to change the property value, and all references of the property now point to the new location.

In this case, you set the property by opening the environment, clicking its **Configuration** tab, and then clicking **Environment Properties**, as shown in the following figure. Components, applications, and processes have equivalent properties tabs.

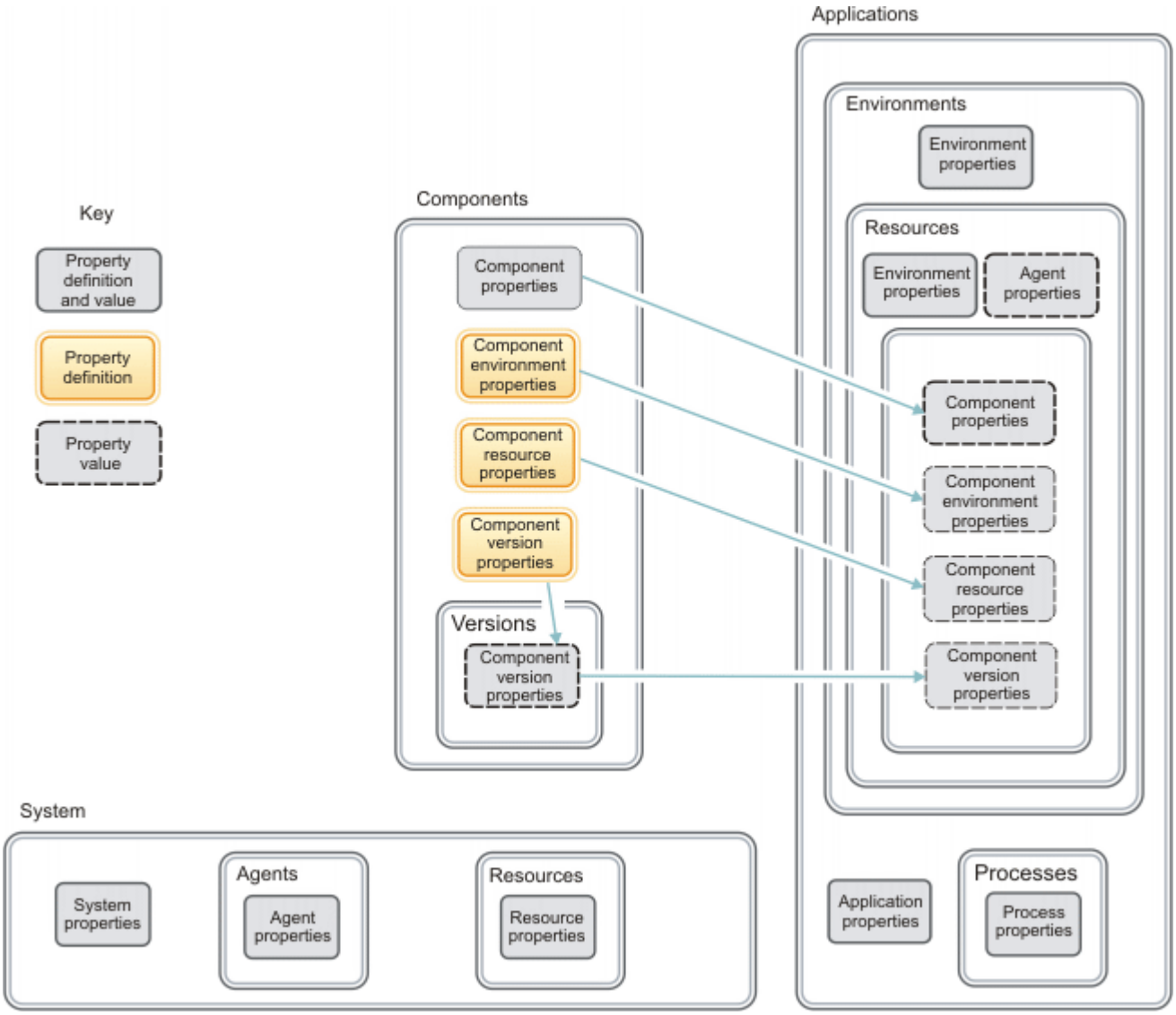


You can then refer to the properties in processes. For example, the following step is in a component process that accesses the database. In one of the step parameters, the value `${p:environment/db.url}` represents the database location.



Properties have scope, which limits where they can be accessed and which properties take precedence over others with the same name. For example, environment properties can be accessed from within the environment, including from components and applications that are deployed to that environment. For more information on property scope, see Property contexts on page 2508.

The following diagram shows some of the major properties that are available and where they can be set and accessed.



- The box at the bottom of the diagram represents the properties that you can set at the system level or the resource tree level. These properties include system properties, which are available in all contexts. The server also stores properties on agents and resources, and these properties are available in all contexts in which these agents and resources are used.
- The box at the center of the diagram represents the properties that you can set on components. Components have component properties, but components also have several types of properties that are specified in a way that is different from other properties. These properties are defined on the component, but their values are set in other places:

  Component environment properties

  Component environment properties are defined on components, but they are not given a value until the component is deployed to an environment. Then, the property behaves much like an environment property; it is available within the scope of the environment.

Component resource properties

Component resource properties are defined on components, but they are not given a value until the component is added to an environment as a resource. Then, the property behaves much like a resource property. Because a component can be added to an environment multiple times, each instance of the component can have a different value for the component resource property.

Component version properties

Component version properties are defined on components, but they are not given a value until you import a component version. Thus, each component version has a different value for the component version property. When you deploy the component to an environment, the component version property behaves much like an ordinary component property.

The boxes at right show which properties are defined in applications and environments, and which properties are accessible in applications and environments. Applications, environments, and the processes and resources in those applications and environments can all have properties. Also, with environments, you can access the properties of components that are deployed to those environments.

You set properties on these HCL DevOps Deploy (Deploy) elements to achieve four primary goals:

- Providing information that varies in different contexts on page 2503
- Storing secure information on page 2503
- Providing property values to other processes, resources, and steps on page 2504
- Setting system variables on page 2504

Providing information that varies in different contexts

Environments are configured differently, which can make automating deployments difficult. With properties, however, you can provide the correct information to each set of components in each environment.

An application might be deployed to several target environments. Each of these environments might be configured differently; for example, tools and resources might be in different directories, or the application must use different ports depending on the environment. Environment properties provide information about those different environments. For example, an environment property can map to different ports as required by the target computers for applications that run on different ports and that communicate with different databases.

As described in Property contexts on page 2508, you can also use properties to override defaults in certain areas.

Storing secure information

To deploy applications, installing components on a target computer might require IDs and passwords that must remain secure. The user interface might display this information, and testing and debugging might require log reviews. In this case, use secure properties to store the passwords. These properties can contain the ID and password information, but they display in the user interface and in logs as asterisks (*).

Providing property values to other processes, resources, and steps

In complex processes and deployment scenarios, you want to automate as much as possible. Passing property values among processes, resources, and steps helps to limit the requirement to provide input manually and to maintain accuracy and reliability. For example, you can refer to output properties from one step in another step. See Output properties on page 2516.

Setting system variables

Properties are flexible, and you can apply them to processes, environments, components, and even the system. However, keeping track of what properties are set where and on what element can present a challenge. One way to meet that challenge is to set system properties that apply to all situations. Of course, you must be judicious in setting system properties so that you can be flexible in applying properties to other elements. When you want to define a property one time and have it apply to all situations, the system property reduces set up time.

Assume that you have one Maven repository for all your code. You might refer to the address for this repository many times in many different applications. You don't need to set the address for this repository more than once. You set a system property, for example: `maven.repository.url=/theURL/`. In most places in a deployment process, use this property in place of the actual address. If the address of the Maven repository changes, you can universally change all references by changing the system property.

Creating, setting, and editing properties

Properties can be created for most elements, including components, resources, applications, and agents. You can set and change properties in many different ways.

About this task

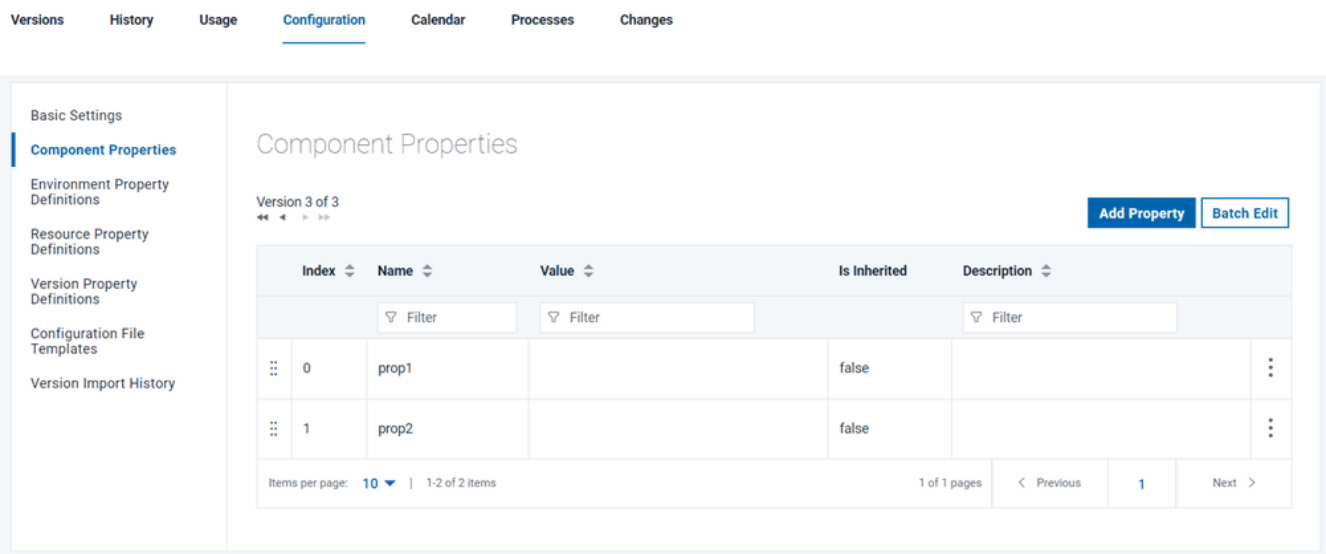
A property represents a user-defined parameter. Properties are typically accessed in processes. For information about using properties, see Referring to properties on page 2509.

Setting properties manually

You can create and edit properties on many elements by opening the element and going to its **Configuration** tab.

About this task

To edit most of the properties for an element, open the element, go to its **Configuration** tab (not the **Configuration** tab at the top of the page), and then click ***element_type* Properties**. For example, to edit component properties, click the component, click **Configuration**, and then click **Component Properties**, as shown in the figure.



However, if you created an application from a template, any environment or application properties that you specified are not available on the **Configuration** tabs. You can view and edit, but not delete, these properties by clicking the element's **Configuration** tab (not the **Configuration** tab at the top of the page), and then clicking **Basic Settings**.

To add a property, click **Add Property**, and provide the information about the property, including its name, value, and description. You can also edit multiple properties at once by clicking **Batch Edit**. Then, in batch mode, specify the properties in name-value pairs that are separated by line breaks, as in the following example:

```
Property1=Value1
Property2=Value2
Property3=Value3
```

Escape backslash, equals, and colon characters (`\`, `=`, `:`) with a backslash, for example:

```
WindowsPath=C\:\\myfolder\\myfile.txt
PropertyWithColons=This string\: has a colon
```

To create a property, put the new property and its value on a new line. To include a line break in a property value, use "`\n`". Be sure to click **Save** to save your changes when you are finished editing multiple properties. To close batch edit mode, use the **Use Table** command. Secured values are encrypted when they are displayed in batch mode. If you modify an encrypted value, the modifications are encrypted when you exit edit mode.

Note: You can reorder the properties by moving them up or down using the row handler ⁘. Property order can only be changed while sorting by index.

Auto-completing properties

Starting in v7.0.5, auto-completing properties were added to the script editor fields. The auto-complete functionality works in application, component, and generic processes.

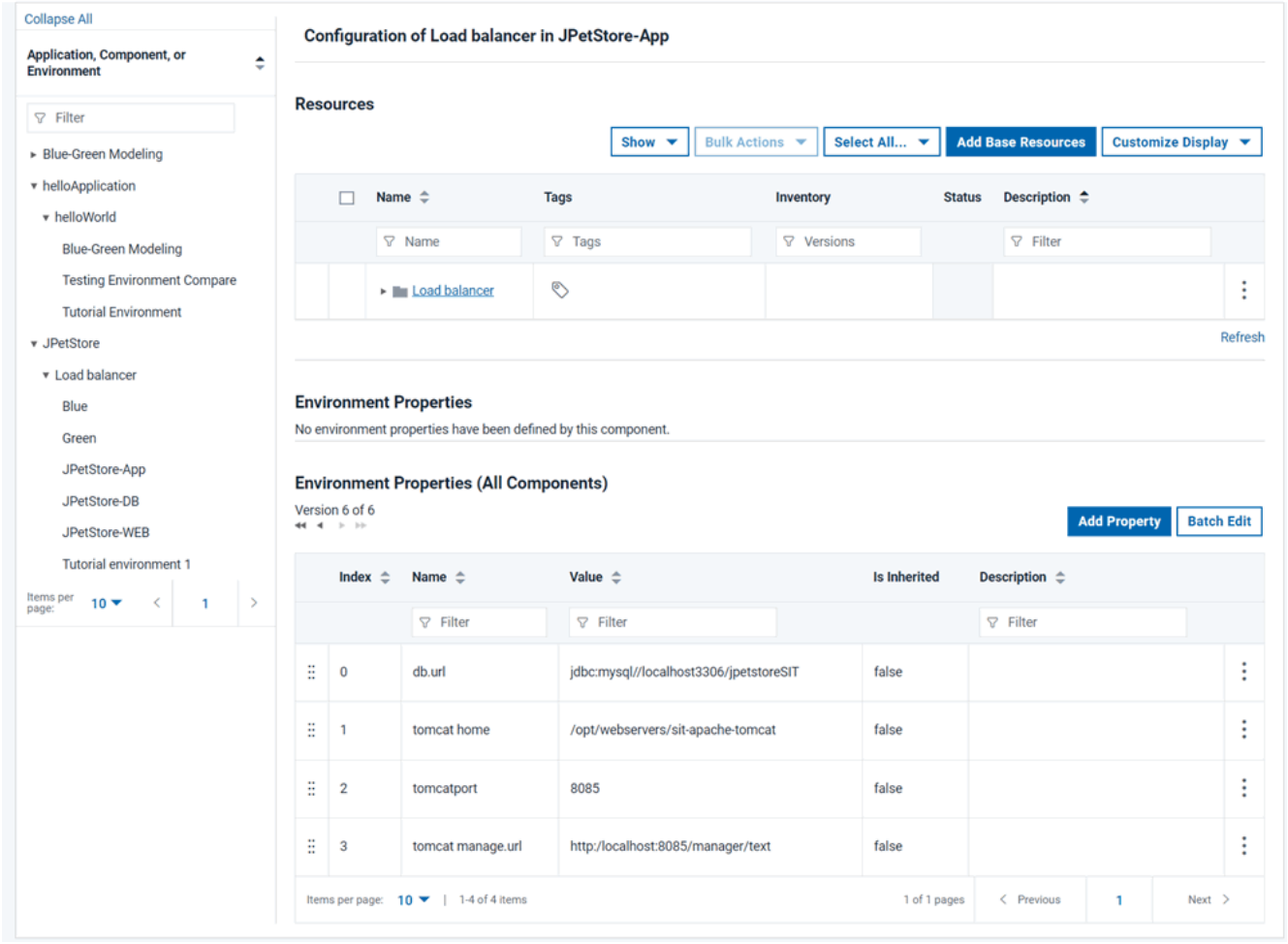
Setting properties with the **Configuration** tab

On the top-level **Configuration** page you can directly manage properties for applications, components, and environments.

About this task

Access the **Configuration** page by clicking the **Configuration** tab. From this page, you can add properties to applications, components, and environments.

Configuration

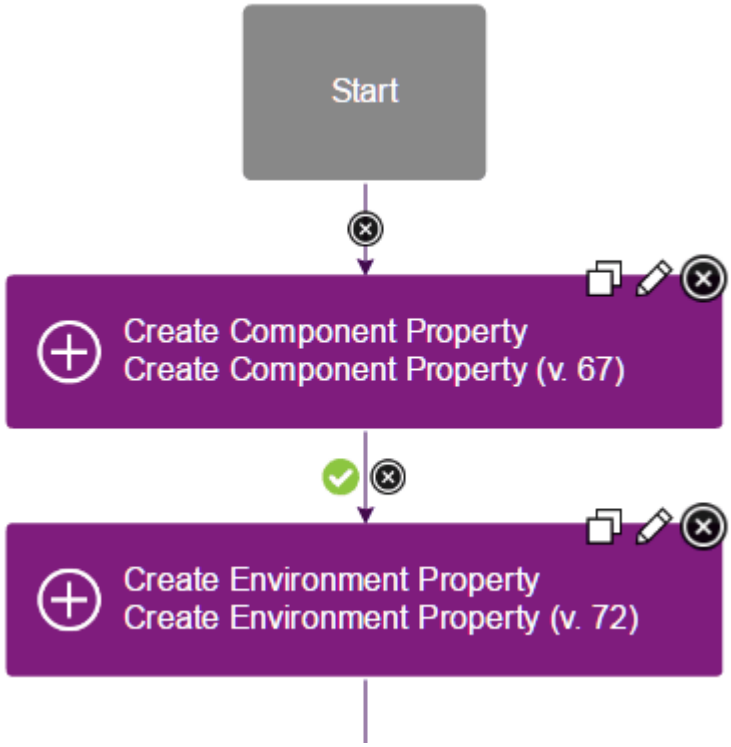


Setting properties in processes

Several different steps are available to create and set properties in processes.

About this task

For example, the Deploy component plug-in contains the Create Component Property step. This step creates a component property or changes the value of an existing component property.



Note: If you use these steps to change the value of a property, the new property value is not available until the next process runs. Later steps in the same process use the same value that the property had when the process started.

Setting properties with the REST API and command-line client

About this task

Several commands in the REST API and in the command-line client can set properties. See REST API reference and Command-line client (CLI) reference on page 2591.

Setting output properties

About this task

To pass properties from one step to another within a process, you can use output properties. See Output properties on page 2516.

Renaming properties

About this task

Properties can be renamed if the new name is not already in use. Renamed property references are not updated automatically.

Setting properties from external files

About this task

You can retrieve property values from an external file. See Retrieving property values from external files.

Property contexts

Properties are available in various contexts. Not all properties are available for all contexts. To plan and set properties, you must know where to access properties and how that setting behaves in context.

The following table shows the contexts in which properties are available and how you can use the properties.

Tip: The context that you set a property in can have narrow or widespread influence. For example, if you set a system property, that property persists in all areas. The key to setting properties effectively is to find a balance to setting narrowly as possible without setting them so narrow that you create extra work for yourself.

Example: Overriding server locations Consider this example: Initially you have one Tomcat server. To store the location of that server, you create an application property with the name `Tomcat.server.url` and the location of that server. Your environments refer to that property to know where the Tomcat server is. Then, suppose that you add a second Tomcat server. This Tomcat server is intended for use only by specific environments. So on those specific environments, you create an environment property with the same name `Tomcat.server.url` and the location of the second server. Now the environment has two properties with the same name: one system property and one environment property. The order of precedence determines which property value is used. Environment properties come before system properties in the order of precedence. Therefore, the environment uses the value of the environment property instead of the system property. In this way, you can use system properties to set defaults and then you can override those properties in specific contexts.

The following table shows the order of precedence of properties.

Context	Description
Process	Available on the **Configuration** tab of the process, whether it is a generic process, application process, or component process. Reference syntax: `${p:propertyName}`.
Component version	Available on the component **Configuration** tab. Click **Component > *selected component* > Configuration > Version Property Definitions**. Then, specify values on individual component versions. Referenced: `${p:version/propertyName}`.
Resource	Available on the resource **Configuration** tab. Click **Resources > *selected resource* > Configuration > Resource Properties**. Referenced: `${p:resource/propertyName}`. You can also use the following syntax to get a comma-separated list of all properties: `${p:resource/allProperties}`.

Context	Description
Agent	Available on the agent **Configuration** tab. Click **Resources > Agents > *selected agent* > Configuration > Agent properties**. Referenced: `${p:agent/propertyName}`. In a For Each Agent context, you can use the following syntax to get a comma-separated list of all properties: `${p:iteration/allProperties}`.
Environment	Environment properties are available on the **Configuration** tab for environments. Also, components can have component environment properties; these properties transfer to the environment when the component is deployed. An environment property overrides the value that is set on a component environment property that has the same name. Reference syntax: `${p:environment/propertyName}`. You can also use the following syntax to get a comma-separated list of all properties: `${p:environment/allProperties}`. Both the component and environment properties use the same syntax. For example, assume that you are deploying a web application to three environments. Each environment might have the application server in a different location. You can specify this location in an environment property on each environment.
Component	Available on the component **Configuration** tab. Click **Component > *selected component* > Configuration > Component Properties**. Referenced: `${p:component/propertyName}`. You can also use the following syntax to get a comma-separated list of all properties: `${p:component/allProperties}`.
Application	Available on the application **Properties** tab. Click **Applications > *selected application* > Configuration > Application Properties**. Reference syntax: `${p:application/propertyName}`.
System	System (global) properties are available on the **Settings** tab. Click **Settings > Properties**. Reference syntax: `${p:system/propertyName}`. The system property is broadest. If you plan to deploy an application, use application properties because it's more specific. If you have an `xyz` system property and an `xyz` application property, and you want to use the value of the system property, you must provide the entire context. Otherwise, the value resolves to the application property. You can also use the following syntax to get a comma-separated list of all properties: `${p:system/allProperties}`.

Note: The system properties are only available during the process executions. Do not use them in your System Setting configurations.

Referring to properties

HCL DevOps Deploy (Deploy) provides several ways to refer to properties.

Referring to properties

Normally, properties can be referred to without scope; but some properties, such as stepoutput properties, are referred to by scope. The scope is the type of artifact that contains the property, such as the environment or component. You refer to a property by scope this way:

```
${p:scope/propertyName}
```

You refer to a property without scope this way:

```
${p:propertyName}
```

For example, if you create an environment variable that is named UAT, you can refer to it this way:

```
${p:environment/UAT}
```

You can refer to the same environment variable without scope this way:

```
${p:UAT}
```

In the same way, you can refer to many types of properties, such as application properties, system properties, and resource properties.

Warning: In some cases, you can refer to a property without the `p:` prefix, such as in the code `${myProperty}`. This method for referring to properties is deprecated and should not be used. Referring to properties in this way can conflict with shell script variables or other values.

Use the `?` syntax if you are not sure about a property name:

```
${p?:propertyName}
```

This syntax returns a blank if the property is not found, and avoids the undefined property error.

Properties in application processes and component processes

When you create or edit process steps for applications and components, you can use the autocomplete feature to determine which properties are available in a particular context. When you are editing an input field for a process step, if you type `${p:`, a list of the available property scopes is displayed. Select a scope from the list. The list of all available properties in that scope is then displayed. Select the property to use from the list.

Secure properties

When you create a property, you can specify whether it is secure. Secure properties are stored in encrypted form. Secure properties are encrypted with an Advanced Encryption Standard (AES) algorithm and a 128-bit key. Secure properties are displayed in obscured form in the user interface.

If you create a custom plug-in that uses secure properties, the property value is displayed in obscured form in the user interface. For example, if your plug-in prints the property value to standard output and you examine the standard output in the user interface, the property value is displayed in obscured form.

If your custom plug-in writes the property value to a file, then the property value is not obscured.

Note: Use strong and complex values for secure properties. Because the secure property value is obscured in the output log, any other text that matches the secure property value is also obscured.

Property order of precedence

If a property is defined in multiple places, its value is determined by the *property order of precedence*. The following list defines the order of precedence from highest to lowest:

Process
Component version
Resource
Agent
Environment
Component
Application
System

If you have an environment property that is named `${p:environment/db.hostname}` and a resource property with the same name, you can refer to the resource property by using `${p:db.hostname}` or `${p:resource/db.hostname}`. Because the resource property is higher on the order of precedence than the environment property, in this case you must refer to the environment property by using the scoped format: `${p:environment/db.hostname}`.

Example: Overriding server locations Consider this example: Initially you have one Tomcat server. To store the location of that server, you create an application property with the name `Tomcat.server.url` and the location of that server. Your environments refer to that property to know where the Tomcat server is. Then, suppose that you add a second Tomcat server. This Tomcat server is intended for use only by specific environments. So on those specific environments, you create an environment property with the same name `Tomcat.server.url` and the location of the second server. Now the environment has two properties with the same name: one system property and one environment property. The order of precedence determines which property value is used. Environment properties come before system properties in the order of precedence. Therefore, the environment uses the value of the environment property instead of the system property. In this way, you can use system properties to set defaults and then you can override those properties in specific contexts.

Example: Defining properties in a step

Consider a process that defines properties in a step.

In this example, create a component process with a **Replace Tokens** step. Edit the step to reference a property list, using the **Property List** field. When the process runs, the normal order of precedence is not used. Instead, the order of properties in the property list is used to replace the tokens.

Example: Component Environment Properties Override Environment Properties

In current precedence model, the environment properties overrides component environment properties with the same name. If there are a large number of components, larger environments, or a combination of the both, this precedence model will require several edits on multiple resources. To reduce this effort, you can make component environment property override environment properties.

In this example, name the component environment property `${p:propName}` as opposed to `${p:environment/propName}`. Enable the **Component Environment Properties Override Environment Properties** settings from the Legacy Configuration Options in the System Settings page. When the process runs, the order of precedence takes place as: Component > Component Environment Properties > Environment.

Setting multiple component properties by environment

When components share a property, you can save time by setting the property value on the environment instead of setting it on every component. To use this feature:

1. Click **Components > *selected component* > Configuration > Environment Property Definitions**.
2. On the **Environment Property Definitions** page, define the property.
3. Repeat the process for each effected component.
4. On the environment that uses the components, click **Configuration > Environment Properties > Set a Single Value**.
5. Set the property value.

Component version properties

Properties can be defined for component versions. Each version can have a unique property value. To use this feature:

1. Click **Components > *selected component* > Configuration > Version Property Definitions**.
2. On the **Version Property Definitions** page, define the property. By default, this value is applied to every version that is created for this component.
3. To override a component version property with a unique value, click **Configuration > Version Properties** for the version.
4. Set the property value.

Using the ? wildcard in property references

The question mark character, `?`, can be used to reference properties. The syntax is:

```
${p?:propertyName}
```

For example:

```
${p?:agent/wsadmin.path}
```

returns the value of the property if it exists, otherwise it returns an empty string.

Escaped characters

Deploy escapes the following characters:

```
\
=
,
```

You must replace the character `\` with `\\`, character `=` with `\=`, and character `,` with `\,`.

Default properties

Many default properties are available for use while a process runs. These properties provide information about the process and the elements that are involved in the process, such as applications, components, and resources.

To see the properties that are available to a process, run the process. When the process is finished, go to the **Properties** tab of the process request. This tab lists all the properties that are available to the process.

The following table lists properties that are automatically available to processes and the types of processes that can access each property.

Property	Description	Process types
agent.id	The unique ID of the agent.	Generic processes and component processes
agent.name	The name of the agent.	Generic processes and component processes
application.id	The unique ID of the application.	Application processes, component processes
application.name	The name of the application.	Application processes, component processes
applicationProcess.id	The unique ID of the application process.	Application processes
applicationProcess.name	The name of the application process.	Application processes
component.id	The unique ID of the component.	Application processes, component processes
component.name	The name of the component.	Application processes, component processes
componentProcess.defaultWorkDir	The default working directory for the component. The value of this property is equivalent to `${p:resource/work.dir}/${p:component.name}`, where `${p:resource/work.dir}` is the	Component processes

Property	Description	Process types
	default working directory for the agent and `${p:component.name}` is the component name.	
componentProcess.id	The ID of the component process.	Component processes
componentProcess.name	The name of the component process.	Component processes
currentVersion.id	The ID of the component version that is used in the process.	Component processes
currentVersion.name	The name of the component version that is currently in inventory.	Component processes
currentVersion.type	Whether the component version is a full version or incremental version.	Component processes
environment.id	The unique ID of the environment.	Application processes, component processes
environment.name	The name of the environment.	Application processes, component processes
finalStatus	Whether the process succeeded (`Success`) or failed (`Failure`).	Application processes, generic processes, component processes
parentRequest.id	The ID of the parent process.	Application processes, component processes
previousVersion.description	The description of the version that was previously in inventory.	Component processes that take versions
previousVersion.id	The unique ID of the component version that was previously in inventory.	Component processes that take versions
previousVersion.name	The name of the component version that was previously in inventory.	Component processes that take versions
previousVersion.type	Type of the version as specified on creation that was previously in inventory.	Component processes that take versions
request.id	The unique ID of the process request.	Application processes, generic processes, component processes
request.user.displayName	The name of the user that requested the process.	Application processes, generic processes, component processes
request.user.email	The email ID of the user that requested the process.	Application processes, generic processes, component processes

Property	Description	Process types
request.user.id	The unique ID of the user that requested the process.	Application processes, generic processes, component processes
request.user.name	The user name of the user that requested the process.	Application processes, generic processes, component processes
resource	The unique ID of the resource that the process runs on.	Generic processes
resource.id	The unique ID of the resource that the process runs on.	Generic processes, component processes
resource.name	The name of the resource that the process runs on.	Generic processes, component processes
resource.parent.path	The parent of the resource that the process runs on	Generic processes, component processes
resource.path	The location of the resource in the resource tree.	Generic processes, component processes
resource/system/work.dir	The working directory on the agent resource.	Generic processes
resource/work.dir	The working directory on the agent resource.	Generic processes
server.url	The URL of the HCL DevOps Deploy (Deploy) server.	Application processes, component processes, generic processes
snapshot.id	The unique ID of the snapshot that was used to specify the component versions. This snapshot is sent only when a snapshot is being deployed.	Application processes
snapshot.name	The name of the snapshot that was used to specify the component versions. This snapshot is sent only when a snapshot is being deployed.	Application processes
version.description	The description of the version.	Component processes that take versions
version.id	The unique ID of the component version.	The built-in version of this property is available only to component processes that are not of the Operation(no version required) type.
version.type	Type of the version as specified on creation. FULL or INCREMENTAL	Component processes that take versions.

Property	Description	Process types
version.name	The name of the component version.	The built-in version of this property is available only to component processes that are not of the Operation(no version required) type.
working.dir	The working directory for the process.	Generic processes

Steps in component processes also have access to the output properties of previous steps, as in this table:

Property	Description
stepName/exitCode	The exit code of the component process step. 0 indicates success; any other value indicates failure.
stepName/LOI	The lines of interest (LOIs) are lines of code that the postprocessing script finds in the step's output log. See The post-processing element on page 2576.
stepName/Status	Whether the step succeeded (`Success`) or failed (`Failure`).
stepName/*outputProperty*	An output property from a previous step. See Examples of post-processing scripts on page 2417.

For example, if a prior step in the process is named `step1`, later steps can access properties such as `step1/exitCode`. If the step `step1` creates an output property that is named `output1`, later steps can access that property with the code `step1/output1`.

Output properties

Output properties are specialized properties that process steps generate. You use output properties to pass parameters to subsequent steps in a process.

You can use output properties in component processes and generic processes.

In the JavaScript™ code for your postprocessing steps, you can establish output properties. See Examples of post-processing scripts on page 2417. You set an output property for a step by assigning it a name and a value in the postprocessing script. For example, the following code from the Check If BLA Exists step in the IBM® WebSphere® Application Server Deployment plug-in establishes three possible values for the `blaExists` output property:

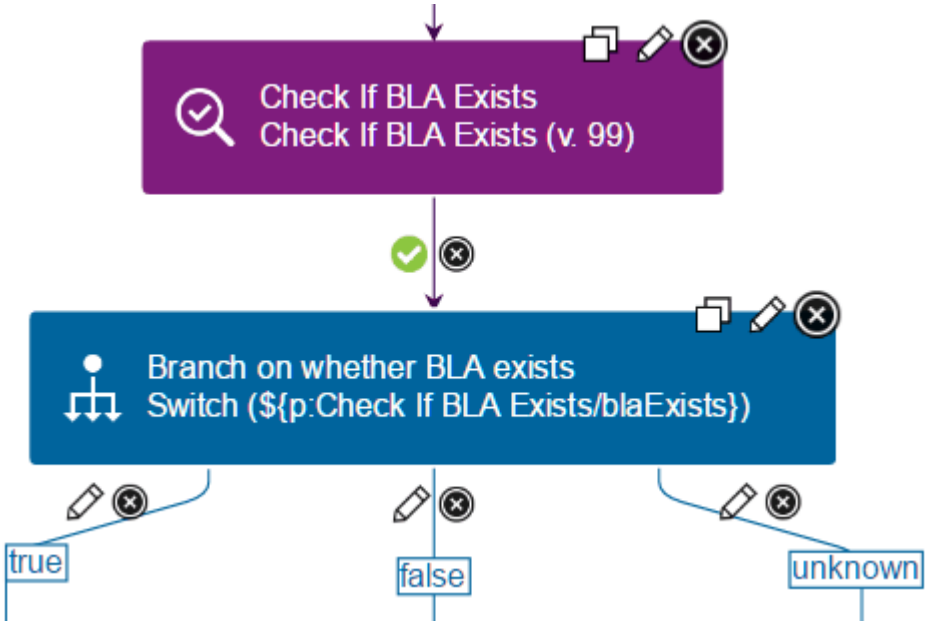
```
def exitCodeChecker = { exitCode ->
    if (exitCode == 0) {
        properties.setProperty("blaExists", "true");
        exitValue = 0;
    }
    else if (exitCode == 30) {
        properties.setProperty("blaExists", "false");
```

```
        exitValue = 0;
    }
    else {
        properties.setProperty("blaExists", "unknown");
        exitValue = 1;
    }
```

In the same way, you can create your own output properties by adding them in post-processing scripts. Use the format `properties.setProperty("propName",propValue);`, where `propName` is the name of the property and `propValue` is the value of the property. For example:

```
properties.setProperty("url", "http://example.com");
```

Then, you can refer to these output properties in other steps in a process. To do so, you must know the name of the step that contains the output property that you use. For example, your process might require different actions if a BLA exists. In this case, you could use a Switch step to branch the process for each case. In the following figure, the Switch step checks the value o f the `blaExists` output property by using the code `${p:Check If BLA Exists/blaExists}`:



You can use output properties in other postprocessing scripts, step preconditions, step properties, or the code of steps themselves. Passing property values among processes, resources, and steps helps limit the requirement to provide input manually and to maintain accuracy and reliability.

Secure output properties

You can set an output property as secure property for a process running on a Web agent. As an example, you can obsecure the value of a password field in the output log and Web UI. Define the secure output property by adding `.$secure` at the end of the property.

For example, set the property in JavaScript™ as:

```
outProps.setProperty("Password.$secure", "xyz");
```

or set the same property in Groovy as:

```
outProps.setProperty("Password.\$secure", "xyz");
```

Note that the agent must have support for the special handling of the indicated output property naming convention. The environment variable `UCD_USE_ENCRYPTED_PROPERTIES` with a value of `true` indicates the support of this functionality in a plug-in step and is handled by the agent.

If the environment variable is not available with the agent, then upgrade the agent to a newer version that has the functionality. Manually adding the environment variable does not enable the desired functionality in an older agent. It rather sets the propery as a regular output property when it is intended to be set as secure.

Manual task output properties

The output properties when added to a process containing a manual task gives you information about the approver and aprroval date. You can view the following information in the output log for a manual task:

- Name of the approver.
- Display name of the approver.
- Date and time of approval in the Epoch format.

  Epoch time is a standard machine-readable format. The `ApprovalTime` property is used to display the time in Epoch format. This format is used internally by another plug-in step.
- Date and time of approval in the UTC format.

  The `FormattedApprovalTime` property is used to get the approval time in UTC format and in a human-readable format.

As an example, you have two steps in a process. The first step is a manual task, named `manualTask` and the second step is a Shell step. You can add the following manual task output properties to the Shell step:

```
echo "${p:manualTask/ApproverName}"
echo "${p:manualTask/ApproverDisplayName}"
echo "${p:manualTask/ApprovalTime}"
echo "${p:manualTask/FormattedApprovalTime}"
```

After the completion of the process, you can view the following information in the output log:

```
=======================
command output:
John P. Doe
John
1123489611901
2024-08-12T19:06:51.901Z
=======================
```

System properties

System properties are global variables for the server.

System properties are available on the **Settings** tab: click **Settings > Properties**.

References to system properties are similar to the following example:

```
${p:system/propertyName}
```

If you create a `SUCCESS` system variable , for example, you would refer to it as in the following example:

```
echo ${p:system/SUCCESS}
```

Output in this case:

```
SUCCESS
```

Property availability

The properties that are available to you depends on the context. For example, in a component process, you can access the component properties of the current component. In an application process, you can access the application properties and environment properties, but not any component properties. The following table shows the processes in which each type of property is available.

To see the properties that are available to a process, run the process. When the process is finished, go to the **Properties** tab of the process request. This tab lists all the properties that are available to the process. The following table lists types of properties that are available to processes and which processes have access to each type of property.

Property type	Processes that can access the property	Sample syntax
Generic process properties	Generic processes	`${p:ProcessProp1}`
Application process properties	Application processes and component processes	`${p:ProcessProp1}`
Component process properties	Component processes	`${p:ProcessProp1}`
Component version properties	Component processes types that include versions, such as Deployment and Operational (with version)	`${p:version/VersionProp1}`
Resource properties	Component processes and generic processes	`${p:resource/ResourceProp1}`
Environment properties	Component processes and application processes	`${p:environment/EnvProp1}`
Application properties	Application processes and component processes	`${p:application/AppProp1}`

Property type	Processes that can access the property	Sample syntax
Component properties	Component processes	`${p:component/CompProp1}`
System properties	All processes	`${p:system/SystemProp1}`
Request properties	All processes	`${p:request.user.id}`, `${p:request.user.displayName}`, `${p:request.user.name}`, `${p:request.user.email}`
Iteration properties	Processes in the For Each Agent step context	`${p:iteration/agent.name}`, `${p:iteration/agent/AGENT_HOME}`
Agent properties	Generic processes and component processes	`${p:agent/PropName}`
External approval properties	Application processes	`${p:ext/PropName}`
Property Definition		
Environment Property Definitions	Component processes	`${p:environment/`*`propertyName`*`}`
Resource Property Definitions	Component processes	`${p:resource/`*`propertyName`*`}`
Version Property Definitions	Component processes	`${p:version/`*`propertyName`*`}`

Tags

A tag is a user-defined, shared, short label that you use to classify, view, and conduct operations on objects such as applications, components, agents, and resources.

If you manage many objects, you can use tags to organize objects so that the user interface is easier to navigate. You can also use tags to simplify deployment operations. If you use application templates, assign tags to components so that you can easily add components to applications that you create from templates.

Creating tags

You can create the agent, application, component, and resource tags from the **Tags** tab.

Before you begin

Ensure that your role has the **Create Tags** permission enabled.

1. Click **Settings > Tags**.
2. Select a tag type such as **Agent Tags**, **Application Tags**, **Component Tags**, or **Resource Tags** and click **Create Tag**.
3. Configure the tag in the **Create Tag** window:

Parameter	Description
Name	Identifies the tag name and is included in the Tags column when added.
Description	Conveys more information about the tag.
Color	Defines the tag color that is displayed in the UI.

4. Save your changes.
   The tag lists in the Agent Tags page.

   Note: Tags once created can only be edited and not deleted.

5. Alternatively, you can create a tag by navigating to the respective agent, application, component, or resource.
   a. Click **Add Tag** icon, select **Add New Tag**.
   b. Provide the tag details as described in the table above.

   Note: To create a tag on an object, you must have the Edit Basic Settings permission enabled on the object.

Adding tags to objects

Add tags to objects to simplify object management, to improve object sorting, and to facilitate task execution by tag.

Before you begin

Ensure that your role has the **Edit Tags** permission and the **Edit Basic Settings** permission on the object type is enabled.

1. Navigate to the object.
   You can apply tags to applications, components, and resources. You can apply tags to resources in your current resource tree or to resources in a resource template. See Creating resource templates on page 2333.
2. Hover the mouse pointer over the object, and click the **Add Tag** icon that is displayed.
   Result
   A window opens where you can select or create a tag.
3. Expand the list, and select a tag.

What to do next

- To organize a list of objects by tag, click **Flat list**. Select **By Tag** to organize the objects into a hierarchical list that is sorted by tag.
- To tag multiple objects at a time or to remove tags from multiple objects at a time, first select the check boxes to left of the objects. Then, click **Actions**, and select **Add Tag** or **Remove Tag**.
- To view all objects that have the same tag, in the tags field at the top of any list of objects, type the name of a tag, and then press Enter. Only the objects that are tagged with the exact text that you type in the **Tags** field are displayed. If a tag name contains spaces, surround the name with double quotation marks when you type it in the **Tags** field.
- To select all objects that have the same tag, first filter the objects as explained in the previous section. Then, click **Select All**, and select **Matching Filter**.
- To quickly assign a process to a specific agent, first tag the agent. Then, click the top-level **Processes** tab, click a process, and then click **Set** . Filter the agents as explained in the previous section, select the agent, and then click **OK**.
- To limit the installation of multiple components in application process to certain components or resources, first tag the components or resources. In the application process, click and drag the **Install multiple components** step to the process editor. In that step, use the **Component Tag** field to deploy the components that have a specific tag and the **Limit to Tag** field to use resources with a specified tag.
- To automatically import versions of a component by using any one of several agents, tag the agents. From the **Components** tab, click **Create Component**. Click **Import new component versions using any agent with the specified tag.**, and in the **Agent Tag for Version Imports** list, select the tag.

Grouping servers by using tags

You can use tagging to group target servers for organized deployments.

About this task

By using tags and filtering, you can separate resources that represent production servers into two logical groups. If you tag half the resources with a blue tag and half with a green tag, then you can run deployments in a fashion that is sometimes called blue-green deployment. When you plan to deploy a new version of an application to a production environment, deploy it to the blue group first. Meanwhile, the green group of servers still provides the older version of the application. After the new version of the application is deployed to the blue group, repeat the process with the green group. In this way, you can deploy a new version of software while still providing uninterrupted service in your production environment.

1. Tag resources, which represent target servers, in two groups.
   To learn more about tagging, see Adding tags to objects on page 2521. If you have resources that are organized in groups, you can apply a tag to a top-level group.
   Example
   For example, create top-level groups for the Blue servers and the Green servers. If you tag the two top-level groups as `Blue` and `Green`, then all resources that a particular group contains are included in deployment steps when you limit deployment to a specific tag.
2. Create two application deployment processes that correspond to the two tags.
   When you add an `Install Component` step to an application process, you can use the **Limit to Tag** field to restrict the deployment to resources that have a specific tag.
3. Deploy the new version of an application by using the first application deployment process.
   Example

For example, deploy the application to a group of servers that are tagged as `Blue` servers.

4. Check that the new version of the application is running as expected.
5. Deploy the new version of an application by using the other application deployment process.

   Example

   For example, deploy the application to a group of servers that are tagged as `Green` servers.

Component tags and component resource tags

You can tag components while you work on the **Components** tab, or you can tag components while you work in the **Resources** tab. However, the tags that you associate with components in these two contexts behave differently. Furthermore, you can also include components in a resource by using the tags that are associated with components in the **Component** tab.

In the **Components** tab, you can tag components according to your requirements. See Adding tags to objects on page 2521. You can organize and associate components by using tags.

In the **Resources** tab, you can tag components and other objects just as in other tabs. When you tag a component in the **Resources** tab, that tag is not applied to the component throughout all tabs. The component resource tag applies to the component only when the component is related to a resource.

As you develop processes, you can use the component resource tag for a few things. You can specify a set of components to be used for an Install Multiple Components or Rollback Multiple Components step in an application process. Specifying a set of components this way runs a process for each component with the specified component tag in the environment. Alternatively, you can also add a component resource tag to an agent resource in the resource tree. Adding the tag is a way of directing that this is a component slot that can be fulfilled with any component that is tagged with the corresponding component tag.

Tip: In the **Resources** tab, you can click the **Actions** menu, click **Component Tag**, and select the tag of components you want to add to the resource. By adding components this way, you automate the components that are included in resources. Otherwise, you have to add components manually and individually to resources.

Webhooks

Using a webhook, you can send text-based responses to users in messaging platform such as MS Teams and Slack when a specific event occurs during deployment process.

The webhooks allows the processes to submit a web-based notification to other applications such as MS Teams and Slack whenever a specific event occurs. Webhooks are transmitted via HTTPS, usually as a POST request over a specific URL. The POST data is interpreted by the receiving application's API, which triggers the requested action and sends a message to the original application to confirm the event has happened.

Managing webhook template

You can create and manage a webhook template to allow Deploy to send notifications to your messaging platform for a specific event.

Before you begin

Make sure that you have configured the template body for your event. See How-tos on page 1998 for example JSON body for Slack and Teams applications.

1. Click **Settings > Automation > Webhook Template**.
2. Click **Create Webhook Template**.
3. Configure the template in the **Create Webhook Template** window as follows:

Parameter	Description
Name	Specify the webhook template name.
Description	Add additional information about the webhook.
Body	Define the template body in JSON format. To get started with a template format,
Type	Indicate the event type that triggers the notification. Webhooks support events such as Process Success, Process Failure, Process Started, Scheduled Process Not Started, Approval Completed, Approval Failed, Approval created, External Approval Completed, External Approval Failed, and Task Created.
Teams	Allow access of a template type to specific teams.

4. Save your changes.

Results

The Webhook template is created and lists in the **Webhook Templates** page.

Adding webhooks to process

You can add a webhook template to applications and generic processes for the process notifications to appear in your configured Teams or Slack messaging applications.

Before you begin

Before you add a webhook to application, make sure that you have created an environment for the application and have resources added to it.

1. Open the webhook configuration UI using any of the following methods:
   - To add webhook to an application, go to **Applications > *application name* > Configuration > Webhooks > Add Webhook**.
   - To add webhook to a process, go to **Processes > *process name* > Configuration > Webhooks > Add Webhook**.

   The **Add Webhook** window is displayed.
2. Copy the generated webhook URL and paste it in the **URL** field.
   Refer to the Slack or Teams documentation for information about how you can get this URL.
3. If you are using a proxy server for Teams, you must specify the following details: **Proxy Host**, **Proxy Port**, **Proxy Username**, and **Proxy Password**.

These fields can also be automatically populated by setting the corresponding system properties: `proxyHost`, `proxyPort`, `proxyUsername`, and `proxyPassword`.

4. Specify an **Event Type**.
5. Select a template name from the **Webhook Template** field.
   See Managing webhook template on page 2523 for more information.
6. If you are adding the webhook to an application, specify the environment name from the **Environments** field.
7. Submit your changes.

Results

The webhook template is added to the application. When a process runs, the notifications appear in your configured messaging platform app.

Notes:

- Notifications can be out-of-order in any of the following cases:
  - Message A is sent and fails to complete for a reason. For example, the remote server has an error.
  - Message B is sent and completes.
  - The remote server error is now resolved.
  - The message A is resent, and completes.
- When you export or import an application or a generic process, the webhooks are also imported/exported with them.

Deploying

Learn how to deploy applications in HCL DevOps Deploy (Deploy).

Getting started with running deployments

You run a deployment by running an application process in a target environment.

You run a deployment in a specific application environment. When you start a deployment, you identify the application process and the component versions (or snapshot) to deploy. Any process that is defined for the parent application can be selected. All components that are associated with the application can be used. When you select component versions, you can filter the list to identify components of interest to you.

When an application process starts, the **Application Process Request** pane displays information about the application's status and provides links to logs and the application manifest. If the process requires an approval or manual task, you respond on this pane also.

HCL DevOps Deploy (Deploy) provides two different Web UI for starting an application deployment process. One is single-page form (default) and the other is deployment wizard. To use deployment wizard UI, go to application configuration, and enable **Use Deployment Wizard**.

Related topics:

- How to run a deployment; see Deploying applications on page 2526.
- How to create notification schemes; see Creating Notifications in a Notification Scheme on page 2081.
- How to setup authorizations; see Creating an approval process on page 2388.
- How to create application gates; see Environment gates on page 2360.

Deploying applications

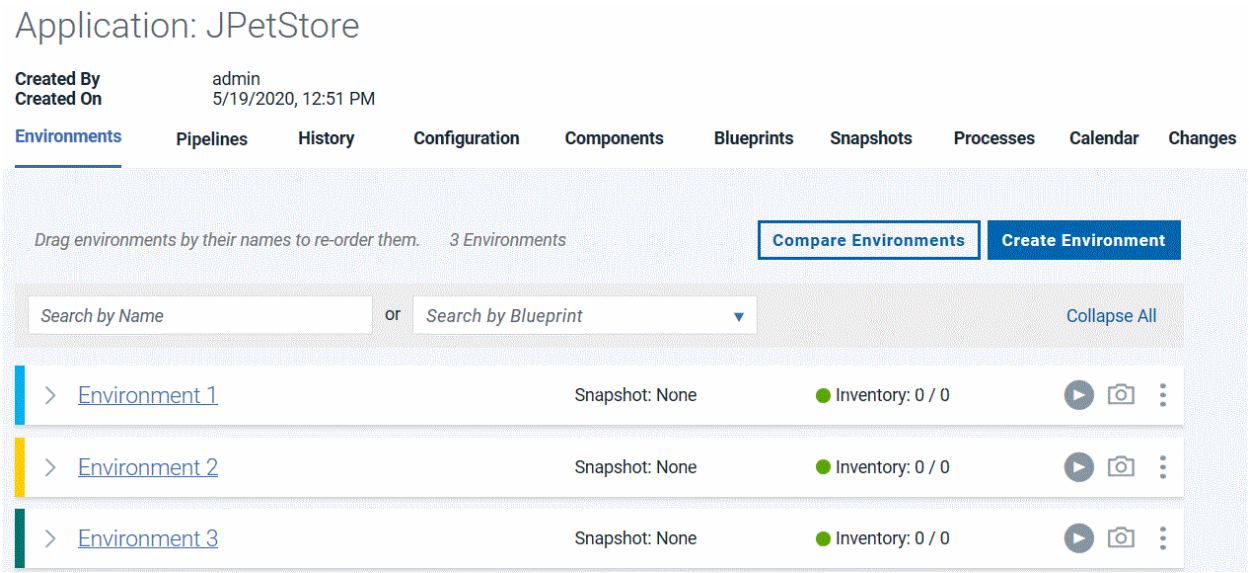
To deploy application components, run an application process on an environment.

Before you begin

Create an application, add components to it, and create an application process that deploys the components.

1. In the web UI, click the **Application** tab.
2. Click the name of the application.

   The application environments are displayed.



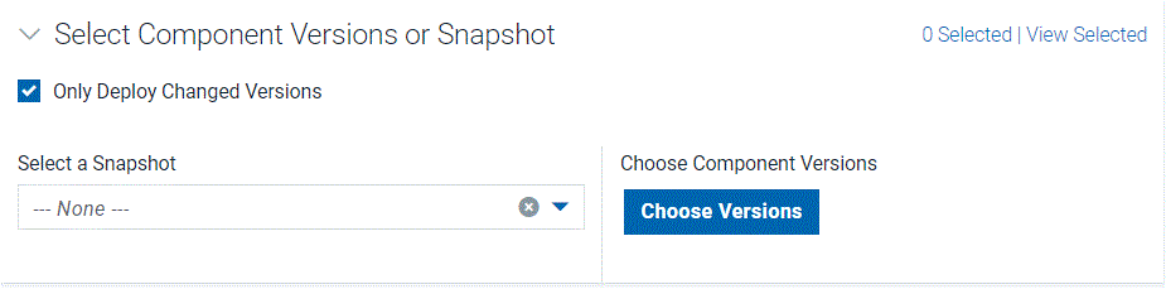
3. For the environment to run the deployment in, click **Request Process** .

   Result

   The **Run Application Process** window is displayed.
4. Select the process to run from the **Process** list.

   All processes for the application are listed.
5. Specify component versions or snapshot to deploy.

a. Expand the **Select Component Versions or Snapshot** area.



b. Enable the **Only Changed Versions** option, if not already, to specify whether to deploy only versions that have changed or to deploy all versions.
This option is enabled by default. If it is enabled, no previously deployed versions are deployed again. Disable the **Only Changed Versions** option to deploy a version regardless of whether it is already deployed. For example, if the inventory is out of date, disable the option.

c. To use a snapshot, select it from the **Select a Snapshot** list.
If you select a snapshot, the deployment automatically uses the component versions that are defined for the snapshot.

d. If you are not using a snapshot, specify the component versions to deploy by clicking **Choose Versions** and specify the versions of each component to deploy.
   - To specify the versions for each component individually, click **Add** next to a component in the table, and select versions to deploy from **By Version Lookup**, and then choose any one of the following criteria:

Option	Description
Latest Available	Selects the most recent component version at the time that you make this selection regardless of when the process runs. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that is available then, click **Latest Available at Execution Time**.
Latest with Status	Selects the most recent component version at the time that you make this selection. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that has the specified status that is available at the later time, click **Latest with Status at Execution Time**.
Latest Available at Execution Time	Selects the latest component version that is available when the process starts. The version that is used might be generated after you make this selection.

Option	Description
Latest with Status at Execution Time	Selects component that have a status that you specify in the next dialog box. The version that is used might be generated after you make this selection.
Current Environment Inventory at Execution Time	Uses the environment inventory that is available when the process starts. The environment inventory might be different from the inventory that is available when you make this selection.

- To select versions for all components at once, click **Select For All**, and then select an option from the list. Version options are described in the following table:

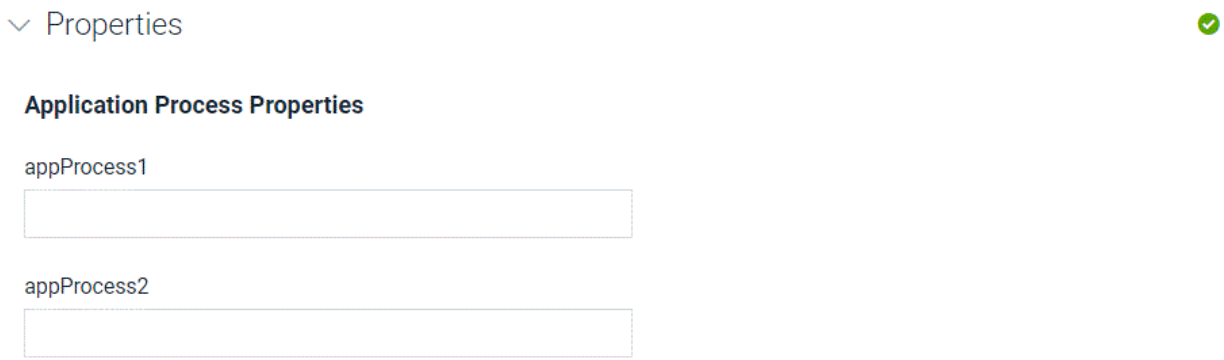
Version option	Description
Latest Available	Select the most recent component version at the time that you make this selection regardless of when the process runs. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that is available then, click **Latest Available at Execution Time**.
Latest with Status	Select the most recent component version at the time that you make this selection. If you schedule a process to run, for example, a week after you make this selection and you want to use the most recent version that has the specified status that is available at the later time, click **Latest with Status at Execution Time**.
Versions With Name	Select the most recent component version whose name matches a string.
Current Environment Inventory	Uses the versions that are listed for that environment when you select it. This selection is typically related to the last version that was deployed to a certain environment of each component. For example, say that you deployed component A, version 1, to your environment. Then, in a separate deployment, you deployed an incremental version 1.1 to the same environment. If you look at the current environment inventory in the **Application** view, you see that the environment contains versions 1 and 1.1. Typically, you don't want to redeploy what is already there. However, this selection is useful when a deployment fails or is modified on the target computer itself, outside of Deploy. So if the deployment of version 1.1 succeeded on 10 agents but failed on two, you can schedule essentially a retry. Use of this selection is not common.

Version option	Description
Latest Available at Execution Time	select the latest component version that is available when the process starts. The version that is used might be generated after you make this selection.
Latest with Status at Execution Time	Selects component that have a status that you specify in the next dialog box. The version might be generated after you make this selection.
Current Environment Inventory at Execution Time	Uses the environment inventory that is available when the process starts. The environment inventory might be different from the inventory that is available when you make this selection.
None (Clear All)	Clears your selection.

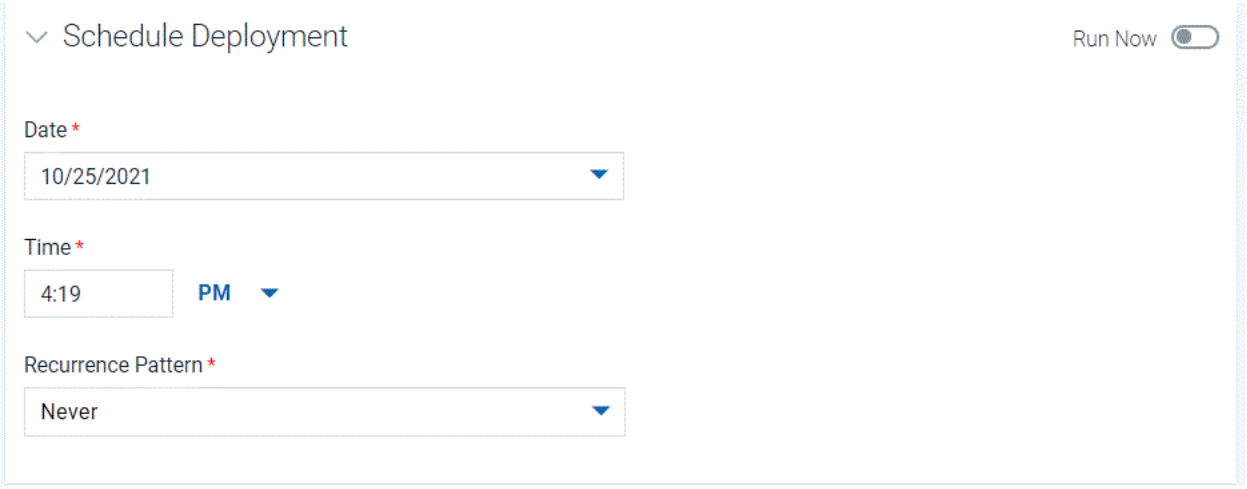
Note: If you select multiple versions for one component, the versions are installed in the order you select.

Note: Make sure to select a version for each component. If you do not select a version for a component, that component is not included in the application process.

6. Expand the Properties area and specify application properties for the deployment, if any.



7. Schedule the deployment for a later date by disabling **Run Now** and then specify the date, time, and recurrence pattern of the deployment.



8. Click **Submit**.

   Result

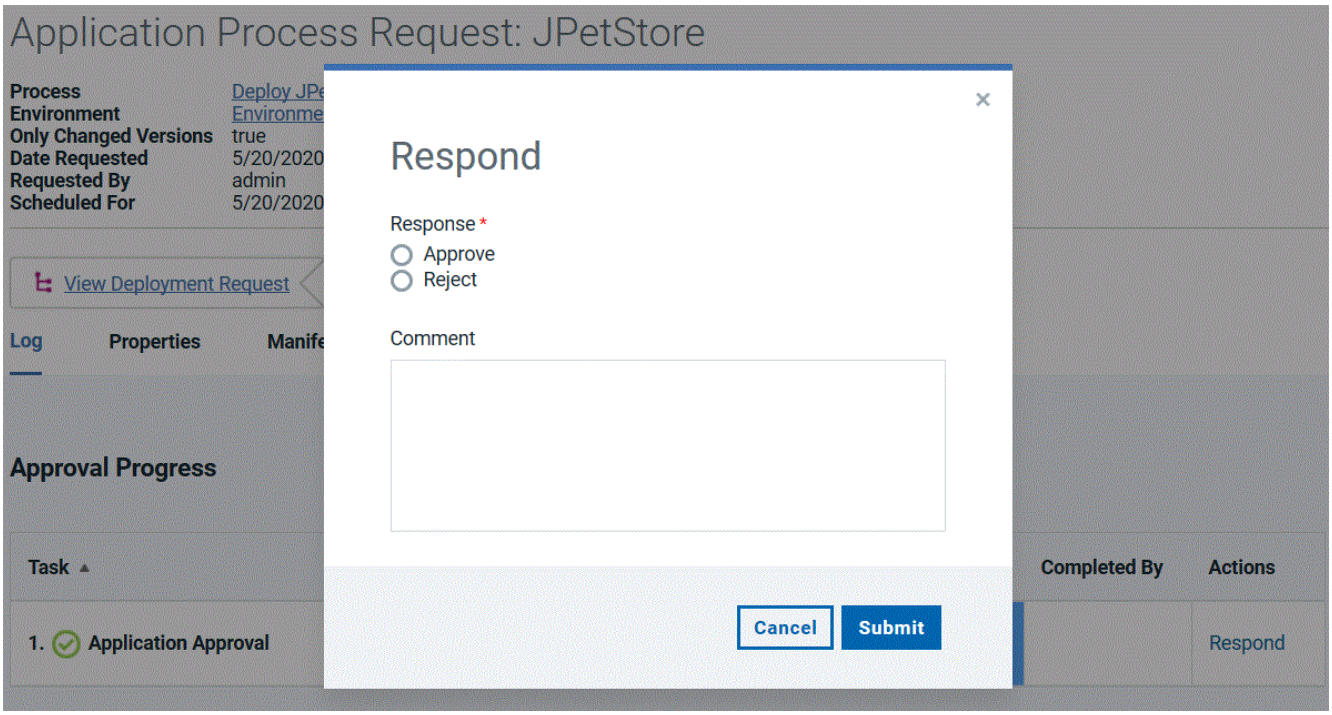
   The application process starts immediately unless it is scheduled for a later time.

Example

For an example, see Deploying a simple web application on page 1934.

What to do next

When a process starts, use the **Application Process Request** pane to review the deployment status. This pane is also used if the process requires approvals, as shown in the following figure.



The application process request shows each step in the process. Within each process, the steps are not shown in the order that they run in. Instead, steps are shown in the order of appearance in the process editor, from the steps at the top of the canvas to the steps at the bottom of the canvas.

The deployment is considered successful when the application process finishes. An application process might skip component processes and still finish. This means that a technically successful application process might not deploy all of the expected components. To ensure that all components deploy successfully, review the deployment details. Expand a process step and click **View Child Execution** to open the **Deployment of Component** pane, which can be used to review the deployment details.

In the **Deployment of Component** pane, you can review the deployment output log, error log, and input and output parameters.

Deploying components to the z/OS® platform

Use the toolkit to deploy to the IBM® z/OS® platform.

Before you begin

Deploying components to the IBM® z/OS® platform is similar to deploying to other platforms. Create an application, components, environments, and resources. Import versions of artifacts to the components, and then create an application process that deploys the components to a target environment.

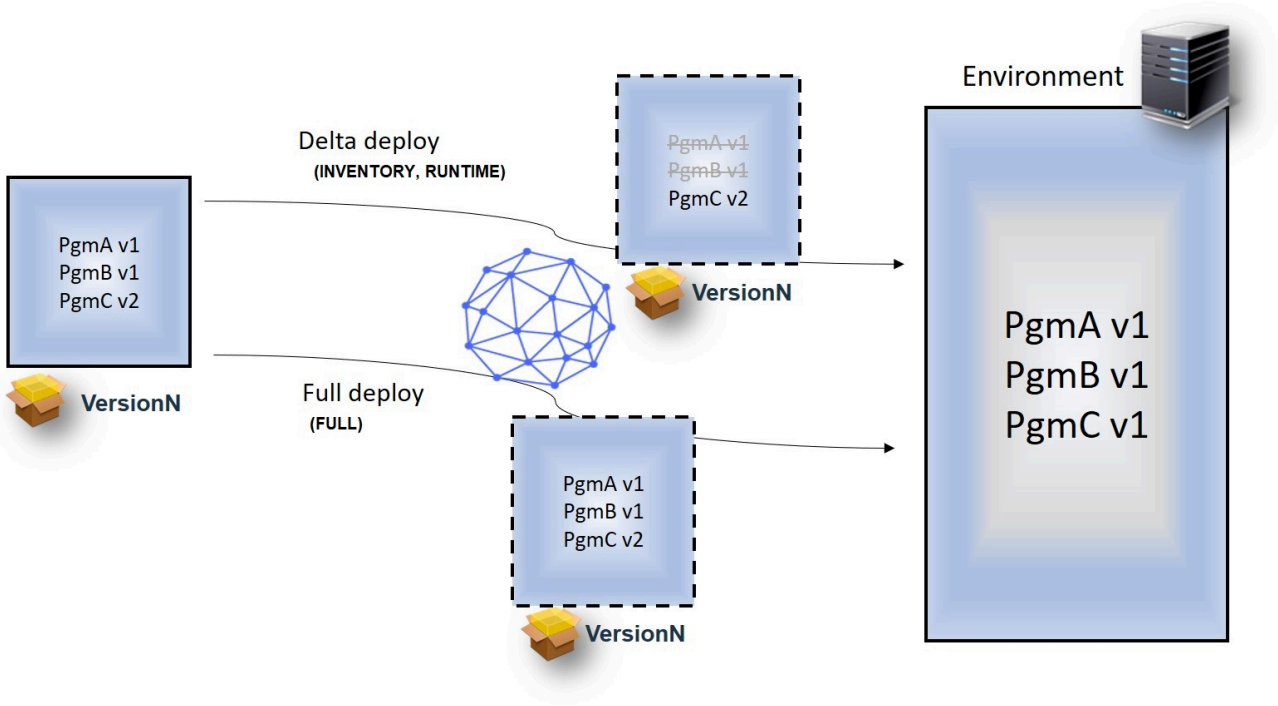
About this task

When you create a component for z/OS artifacts, select **z/OS®** as the component type.

Versions can be imported to a z/OS component by using the z/OS File source configuration plug-in or by using the buztool command line from an z/OS agent.

Two or more z/OS component versions can also be merged from a snapshot into one to simplify the deployment to the next environment. Refer to Merging z/OS component versions on page 2292.

In your application process with the components to deploy, configure the **Deploy Data Set** process step as either a **FULL**, **INVENTORY**, or **RUNTIME** deployment for z/OS data sets and HFS



files.

A **FULL** deployment type replaces all artifacts with artifacts in the current component version.

An **INVENTORY** deployment type, or delta deployment, reduces deployment time significantly by deploying only the changes between artifacts.

A **RUNTIME** deployment type is a delta deployment that uses checksum logic to compare artifacts to be deployed with the same artifacts in the target environment. This check is done for every artifact. Only artifacts with checksums that don't match are considered as changed and used for deployment.

Note: Two artifacts are considered the same when at least one attribute can be used for comparison and all attributes that are used for comparison match exactly.

For each artifact in a delta deployment, the following attributes are compared to the latest inventory version of the same artifact.

Parameter	Where Used
Last Modified Timestamp	**INTENTORYRUNTIME**

Parameter	Where Used
Custom properties starting with **SYS.id** (aka identification properties)	**INVENTORYRUNTIME**
checksum	**RUNTIME**

1. Install the z/OS® agent. See Installing the z/OS agent.
2. Create a ship list file.
   See Ship list files on page 2533.
3. Create a deployable component version.
   See Creating z/OS component versions on page 2279.
4. Create component processes and application processes as you would for any other deployment, by using the z/OS® Utility plug-in for process steps that retrieve and deploy z/OS® artifacts.
   Example
   For example, use Download Artifacts for z/OS process steps to retrieve z/OS® artifacts. Configure the Deploy Data Set process step for either a **FULL**, **INVENTORY** or **RUNTIME** deployment of the z/OS® data sets and HFS files.
   The z/OS® Utility plug-in also includes process steps for rolling back data sets, for running commands from the time sharing option (TSO) command line, and for running ISPF commands. For best results, create components by using the MVSCOMPONENT template which has built-in deploy processes.

What to do next

To learn more about the z/OS® Utility plug-in, see the HCL plug-in site.

Ship list files

The ship list file describes the files from the build to include in the new component version to deploy. Ship list files can also use generic artifacts to describe changes that are not physical files, such as configuration changes.

You must create a ship list file before you run the IBM® z/OS® deployment tools. Typically, you write a script that works with your build engine to create a ship list file from the build output. Ship list files are XML files that contain a list of file specifiers.

Partitioned data sets

Partitioned data set (PDS) members are identified by the **PDS** container type and the **PDSmember** resource type. You can use an asterisk (*) as a wildcard for the resource name, if you want all members in a partitioned data set to be included in a package.

For example, to package the BLZCPBTK member from the BLD.JCL PDS and all of the members in the BLD.LOAD1 PDS and the ORDRSET and RDBKC01 members from the BLD.LOAD2 PDS, the ship list file might contain the following lines:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.JCL" type="PDS">
        <resource name="BLZCPBTK" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD1" type="PDS">
        <resource name="*" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD2" type="PDS">
        <resource name="ORDRSET" type="PDSMember"/>
        <resource name="RDBKC01" type="PDSMember"/>
    </container>
</manifest>
```

Sequential data sets

Sequential data sets are identified by the **sequential** container type.

For example, to package the BLD.DATA1 sequential data set, the ship list file might contain the following lines:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name=" BLD.DATA1" type="sequential">
    </container>
</manifest>
```

HFS files

Hierarchical file system (HFS) files used in z/OS UNIX are identified by the directory container type and the file resource type. Both container name and resource name support the Ant file include pattern. (For more information about Ant, see https://ant.apache.org/.) Each directory container must use the **rootDir** attribute to identify the root directory to resolve files from. Only files in that root directory are considered.

For example, to package `*.wsbind` files and `*.wsdl` files from the `wsbind` directory and the `wsdl` directory within the root directory `/u/dave/cicsfiles`, the ship list file might contain the following lines:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="wsbind" rootDir="/u/dave/cicsfiles" type="directory">
        <resource name="*.wsbind" type="file"/>
    </container>
    <container name="wsdl" rootDir="/u/dave/cicsfiles" type="directory">
        <resource name="*.wsdl" type="file"/>
    </container>
</manifest>
```

To package everything from the root directory `/u/dave/cicsfiles`, the ship list file might contain the following lines:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="." rootDir="/u/dave/cicsfiles" type="directory">
        <resource name="**/*" type="file"/>
    </container>
 </manifest>
```

To deploy HFS files, use the same plug-in steps as you use to deploy data sets. HFS files and data sets can be put into the same version and deployed or rolled back together.

Deployment types

You can set the **deployType** attribute on containers and resources. The **deployType** attribute is used to identify different types of artifacts so that they can be processed differently during a deployment. The following ship list file shows **deployType** attributes that are added to the previous PDS example:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.JCL" type="PDS" deployType="JCL">
        <resource name="BLZCPBTK" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD1" type="PDS" deployType="CICS LOAD">
        <resource name="*" type="PDSMember"/>
    </container>
    <container name="BLD.LOAD2" type="PDS">
        <resource name="ORDRSET" type="PDSMember" deployType="CICS LOAD"/>
        <resource name="RDBKC01" type="PDSMember" deployType="CICS LOAD"/>
    </container>
</manifest>
```

Custom properties

You can add custom properties to containers and resources. The properties are displayed in the **Version Artifacts** view and can be used by plug-in steps. For example, properties can be used by the Generate Artifact Information step in the IBM® z/OS® Utility plug-in. The following ship list file shows an example of custom properties.

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.DBRM" type="PDS">
        <property name="COLLID" value="C001"/>
        <resource name="MOD01" type="PDSMember">
            <property name="devowner" value="Martin"/>
        </resource>
    </container>
</manifest>
```

You can add custom properties to the manifest element. When a property is the child of the manifest element, it is created as a component version property. If the value of the property is a URL, a link is added to the version.

To learn more about using custom properties, see "Using custom properties in deployments" in the z/OS Utility Plug-in.

Deletions

You can mark PDS members, sequential data sets, or HFS files for deletion by specifying the containers inside a **deleted** element. Data sets or HFS files that are marked for deletion are deleted from the target environment when the deployment process runs. The deleted artifacts can be rolled back if the backup option is enabled in the deployment step. If you use the ship list file to delete data sets, you can roll back the deletions as needed.

For example, the following ship list deploys PGM1 and PGM2 and deletes PGM3 from the target environment:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.LOAD1" type="PDS">
        <resource name="PGM1" type="PDSMember"/>
        <resource name="PGM2" type="PDSMember"/>
    </container>
<deleted>
     <container name="BLD.LOAD1" type="PDS">
         <resource name="PGM3" type="PDSMember"/>
 </container>
</deleted>
</manifest>
```

Notes:

- Wildcard (*) is not supported in the resource names for a deleted container.
- Wildcards must not be used for directory or file names.

Generic artifacts

Use generic artifacts to describe changes to deploy that are not physical files. For example, use generic artifacts to deploy system configuration changes or middleware configuration changes that are not associated with standard artifacts. Generic artifacts have the `GENERIC` container type.

For example, the following ship list describes two message queues to deploy to the target environment:

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
<container name="QLOCAL" type="GENERIC" deployType="">
  <resource name="LCQ1">
    <property name="maxdepth" value="5000"/>
  </resource>
  <resource name="LCQ2">
    <property name="maxdepth" value="200000"/>
  </resource>
</container>
</manifest>
```

To deploy generic artifacts, use the Generate Artifact Information step from the z/OS® Utility plug-in to generate commands for deploying the artifacts. Then, use an appropriate command step to run the commands. Steps that you can use to run commands include the Submit Job step, the Shell step, and the TSO/ISPF Command step. Other plug-ins can also use the output of the Generate Artifact Information step to deploy changes.

Source information

You can add source information to PDS members and sequential data sets to build traceability from deployable artifacts to the source code version. The source information is displayed in appropriate views in the HCL DevOps Deploy (Deploy) server.

In following example, the source information in the `<inputs>` and `<input>` elements indicate `BLD.LOAD(ORDRSET)` is compiled from two source code files, a main program ORDERSET and a copybook ORDRDATA. The version of the source code is identified by the version attribute.

```
<?xml version="1.0" encoding="CP037"?>
<manifest type="MANIFEST_SHIPLIST">
    <container name="BLD.LOAD" type="PDS">
        <resource name="ORDRSET" type="PDSMember">
<inputs url="">
<input name="ORDRSET" version="3" compileType="Main" url = ""/>
               <input name="ORDRDATA" version="1" compileType="Copybook" url = ""/>
            </inputs>
 </resource>
    </container>
</manifest>
```

Notes:

- The ship list file must be encoded in CP037 format.
- If a data set member name contains invalid characters, the characters are replaced by % to match any characters. Valid data set member name characters are `0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@#$`.

Generating Ship list for Endevor and Changeman packages

You can generate shiplist.xml using Endevor SCL and Changeman configurations. Generated shiplist.xml can later be used to create component version using buztool.sh.

Generating Ship list from Endevor SCL

Run shiplistgen.sh to generate ship list from an Endevor SCL.

Parameters	Required	Description
-d, -dataset	Yes	MVS Dataset that contains Endevor SCL (or) use '-f' option
-f, -file		HFS File that contains Endevor SCL (or) use '-d' option
-o, -output	No	Ship list file name to be written to. By default, a file 'shiplist.xml' is created in current working directory
-m, -mapping	Yes	Mapping file for Endevor
-verb, -verbose	No	Set to true to display trace log

This tool requires Endevor Package SCL containing only MOVE statements in a Data set or file. Refer to the following sample Endevor SCL:

```
MOVE ELEMENT 'CARP001'
     FROM ENVIRONMENT 'PROD'
          SYSTEM 'PAYROLL'
          SUBSYSTEM 'REPORTS'
          TYPE 'COBOL'
          STAGE NUMBER 1
     OPTIONS CCID REQ#43034
          COMMENT 'MOVE INTO PRODUCTION'
          WITH HISTORY.

     MOVE ELEMENT 'CARPBACH'
     FROM ENVIRONMENT 'PROD'
          SYSTEM 'PAYROLL'
          SUBSYSTEM 'REPORTS'
          TYPE 'DBRM'
          STAGE NUMBER 1
     OPTIONS CCID REQ#43034
          COMMENT 'MOVE INTO PRODUCTION'
          WITH HISTORY.

     MOVE ELEMENT 'SCRN001'
     FROM ENVIRONMENT 'PROD'
          SYSTEM 'PAYMENTS'
          SUBSYSTEM 'FINANCE'
          TYPE 'COPYBOOK'
          STAGE NUMBER 1
     OPTIONS CCID REQ#43034
          COMMENT 'MOVE INTO PRODUCTION'
          WITH HISTORY.
```

This tool also requires a mapping file in JSON format. Assign Data set names DSNAME for each TYPE available in Endevor. For example, A SCL type **COBOL** is assigned to Data set name **TEST.ENDEVOR.COBOL** as shown in below sample mapping JSON:

```
[
    {
        "TYPE": "COBOL",
        "DSNAME": "TEST.ENDEVOR.COBOL"
    },
    {
        "TYPE": "COPYBOOK",
        "DSNAME": "TEST.ENDEVOR.COPYBOOK"
    },
    {
        "TYPE": "DBRM",
        "DSNAME": "TEST.ENDEVOR.DBRM"
    }
]
```

For example, the following command creates a ship list using Endevor package SCL stored in file `/u/user/Sample_SCL.txt` and mapping in `/u/user/Endevor_mappings.json`.

```
/opt/devops-deploy/agent/bin/shiplistgen.sh -f /u/user/Sample_SCL.txt -m /u/user/Endevor_mappings.json
```

The shiplistgen.sh creates a **shiplist.xml** file in the working directory.

Generating shiplist for Changeman packages

HCL DevOps Deploy (Deploy) can create its own versions for equivalent packages in Changeman. This is done using the help of shiplist4changeman.sh shell script that is shipped in the z/OS agent.

The shiplist4changeman.sh accepts the following parameters:

Parameters	Required	Description
-m, -mapping	Yes	A file in USS that can map one of (TEXT, BINARY, EXEC) types to COBOL, JCL and other types in Changeman SCM in key value pairs like below: EXEC=LOAD TEXT=COBOL, COPYBOOK, JCL, PROCLIB BINARY=DBRM Multiple values must be separated by commas.
-p, -package	Yes	Name of the package.
-hlq	Yes	High-level qualifier of datasets for Changeman packages. The HLQ can be like: HLQ.*. PKGNAME.*. LOADLIB HLQ.*. PKGNAME.*. LOAD* HLQ.PKGNAME. LOAD* HLQ.*. PKGNAME.*. JCLLIB HLQ. **. PKGNAME.*. COBOL
-o, -output	No	The output xml file to which the shiplist.xml is generated to. The file need not be present in the system.
-verb, -verbose	No	Set to true to display the trace log.

The shell script is found in `<AGENT_HOME>/bin` and hence must be executed from this directory.

For a package "PKG" containing only JCL members, a high-level qualifier "HLQ ", and mapping file "mappingfile.txt" containing TEXT=JCL and output "shiplist.xml", the command displays as below:

```
/opt/devops-deploy/agent/bin/shiplist4changeman.sh -m /u/user/mappingfile.txt -p PKG -hlq HLQ. **.PKG
 -o /u/user/shiplist.xml
```

The output shiplist.xml will look like below:

```
<?xml version="1.0" encoding="IBM-1047"?><manifest type="MANIFEST_LIST"> <container deployType="TEXT"
name="HLQ.PKG.JCL" type="PDS"> <resource name="*" type="PDSMember"/> </container> </manifest>
```

Deploying by using the Job Monitor

You use the Job Monitor to submit jobs and to monitor the job status in your deployment. You must ensure that you have a Job Monitor task running and that you run the HCL DevOps Deploy (Deploy) process to submit or to monitor the job. Then, you must set up Job Monitor connection properties in your HCL DevOps Deploy (Deploy) environment.

About this task

Install the following agent or product and use the Job Monitor that is bundled with it:

- z/OS agent
- IBM Developer for IBM Z
- IBM Rational Team Concert

Note: For the z/OS agent, the Job Monitor is installed automatically in the `hlq.SBUZAUTH` data set. Also, the **LOOPBACK_ONLY** property must be set to OFF in the `Job Monitor` configuration file.

`The hlq.SBUZSAMP(BUZJJCL)` member has the sample PROC for starting JMON. The `hlq.SBUZSAMP(BUZJCNFG)` member is the JMON configuration file that is referred to in the PROC. JMON installed for another product (for example, IDz or IBM RTC) can be used, but the **LOOPBACK_ONLY** property must be OFF

Submitting and monitoring jobs in deployment processes

1. Start the Job Monitor task and check that a task that is called **JMON** is running.
2. To submit or to monitor jobs, run the process that uses the **Submit Job** or **Wait For Job** steps.
3. Set the following Job Monitor connection properties in the Deploy environment or resource:

```
jes.host=localhost
jes.user=z/OS user ID
jes.password=Password of the user. Set blank to use PassTicket authentication.
jes.monitor.port=The port number for Job Monitor host server.
The default port is 6715.
```

Authenticating with a user ID and Password

To use a password to authenticate, store the password in the *jes.password* property.

Note: As you complete the **Submit Job** or **Wait For Job** step, do not store the password directly in the **Password** field. The **Password** field contains plain text only and this field must be used to reference a property.

Authenticating with PassTickets

If you do not provide a password, the plug-in uses PassTickets authentication. You can use PassTickets authentication to eliminate the requirement to store user passwords in Deploy.

To configure your system to use PassTickets, complete the following steps:

1. Activate the RACF PTKTDATA class if it is not already active. The following code shows sample RACF commands:

```
SETROPTS GENERIC(PTKTDATA) SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.USER1 UACC(NONE)
```

2. To create a PTKTDATA profile that defines the secret key and the application name to which it applies for the Job Monitor, complete the following steps:
    a. Define the application name as FEKAPPL.
    b. Define the key as a 64-bit number (16 hex characters).

c. Replace the key16 placeholder with a user-supplied 16 character hex string (characters 0-9 and A-F) as shown in the following sample RACF commands:

```
RDEFINE PTKTDATA FEKAPPL UACC(NONE) SSIGNON(KEYMASKED(key16)) APPLDATA('NO REPLAY PROTECTION - DO
 NOT CHANGE') DATA('URBANCODE DEPLOY')
```

The following example shows the command with the key16 value replaced:

```
RDEFINE PTKTDATA FEKAPPL UACC(NONE) - DATA('URBANCODE DEPLOY') - APPLDATA('NO REPLAY PROTECTION -
 DO NOT CHANGE') - SSIGNON(KEYMASKED(0123456789ABCDEF))
```

d. Read the following scenarios:

i. If the PTKTDATA class is already defined, verify that it is defined as a generic class before you create the profile that was shown previously. The support for generic characters in the PTKTDATA class is new in the z/OS V1.7 release, with the introduction of a Java interface to PassTickets.

ii. If a cryptographic product is installed and available on the system, you can encrypt the secured sign-on application key for added protection. Use the KEYENCRYPTED keyword instead of the KEYMASKED keyword. For more information, see the Security Server RACF Security Administrator's Guide (SA22-7683).

iii. If the Rational Developer for System z or the Rational Team Concert server components are already installed on the system, the PTKTDATA profile might be defined already.

3. To define a profile to generate a PassTicket, you must define the IRRPTAUTH profile in the PTKTDATA class. Then, this profile controls for which user ID a PassTicket is generated, as shown here:

Operation	Profile name	Required access
Generate PassTicket	`IRRPTAUTH.application.target-userid`	Update

4. To set permissions for the Deploy agent to generate a PassTicket so that the agent generates a PassTicket, the user ID of the agent or the impersonated ID must have UPDATE access in the PTKTDATA profile that you created in the previous step. For more information, see the following RACF commands:

```
PERMIT IRRPTAUTH.FEKAPPL.USER1 CLASS(PTKTDATA) ID(AGNTUSR) ACCESS(UPDATE)
```

5. To refresh the PTKTDATA class for the new profiles and permissions to take effect, see the following commands:

```
SETROPTS RACLIST (PTKTDATA) REFRESH
```

Example

See the following PassTicket examples:

Example 1. Agent is started by user AGNTUSR. In a deployment process, a Job needs to be submitted on behalf of user USER1.

```
RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.USER1 UACC(NONE)
PERMIT IRRPTAUTH.FEKAPPL.USER1 CLASS(PTKTDATA) ID(AGNTUSR) ACCESS(UPDATE)
SETROPTS RACLIST (PTKTDATA) REFRESH
```

Example 2. Agent is started by user AGNTUSR. Allow this agent to submit jobs on behalf of any user.

```
RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.* UACC(NONE)
PERMIT IRRPTAUTH.FEKAPPL.* CLASS(PTKTDATA) ID(AGNTUSR) ACCESS(UPDATE)
SETROPTS RACLIST (PTKTDATA) REFRESH
```

Example 3. Agent is started by user AGNTUSR. Allow this agent to submit job on behalf of user AGNTUSR:

```
RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.AGNTUSR UACC(NONE)
PERMIT IRRPTAUTH.FEKAPPL.AGNTUSR CLASS(PTKTDATA) ID(AGNTUSR) ACCESS(UPDATE)
SETROPTS RACLIST (PTKTDATA) REFRESH
```

What to do next

More JES security considerations are described in the IBM Developer for IBM Z documentation. For more information, see JES Security.

Authenticating with Trusted IP

Starting with v2.9.17 or higher, JMON can authenticate client IP as a trusted source and process client requests. This allows the `Wait for Job` and `Submit Job` plug-in steps to run with Agent User Id or Impersonation Id. The User Id and password/ pass-ticket that are usually needed for these steps are ignored and hence can be left empty. If the client (z/OS UCD Agent in this case) is on the same system as JMON then no changes are required. But if JMON is active on another system in the SYSPLEX, TCPIP RACF profiles are required.

While it is recommended to execute JMON as a started task, it is also possible to execute it as a user job. When not active as a started task, JMON will query your security product for explicit permission to start.

Security profile	Required access
BUZ.START.BATCH.*jobName.port*	READ

The security class where this profile resides can be specified with *SAF_CLASS* variable in the BUZJCNFG configuration file and is FACILITY by default. When the profile is not defined, or the class is not active, permission is denied, and batch startup will fail.

Use the following RACF commands to allow USER ID (IBMUSER in this case) to start JMON in batch with job name JMONJCL and port 6715:

```
RDEFINE FACILITY BUZ.START.BATCH.JMONJCL.6715 UACC(NONE)
DATA('Start JMON in Batch')
PERMIT BUZ.START.BATCH.JMONJCL.6715 CLASS(FACILITY) ACCESS(READ)
ID(IBMUSER)
SETROPTS RACLIST(FACILITY) REFRESH
```

Job Monitor customization

The Job Monitor binary is installed with the IBM® z/OS® deployment tools in the hlq.SBUZAUTH data set. You need the assistance of a security administrator and a TCP/IP administrator to complete this customization task.

Before you begin

1. Create a JCL file in a partitioned data set and base the file on the sample in the *high_level_qualifier*.SBUZSAMP(BUZRJCL) file.
2. Use the buztool createzosversion command to create the component version.

Table 761. Arguments for the buztool createzosversion command

Parameter	Required	Description
-c	true	The name of the component in HCL DevOps Deploy (Deploy). The component name can contain only letters, numbers, and spaces.
-v	false	The name of the version to create. If a version is not specified, a version name is generated from the current time stamp. The version name can contain only letters, numbers, and spaces. **Note:** The version name must not end in a space.
-s	true	The location of the ship list file.
-verb	false	To display a trace log, set this parameter to `true`.

Example

For example, the following command creates a new version in the CICSModules component:

```
buztool "createzosversion" "-c" "CICSModules" "-v" "fixbug100" "-s" "/u/ucduser/build/shiplist.xml"
```

As shown, you must enclose all arguments in double quotation marks.

3. Modify the sample file to reflect your system configuration.
4. Submit the JCL.

Results

The content that the ship list file specifies is available in a component in Deploy.

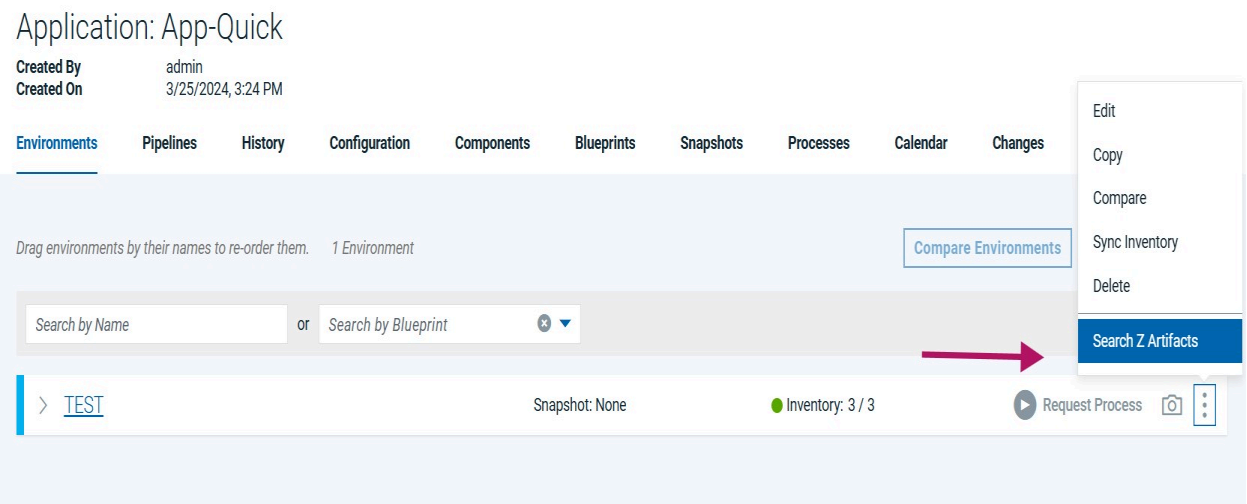
Searching the z/OS environment inventory

You can search z/OS environment inventory for any application that contains a z/OS type component. Search is performed for a specific file in the artifacts belonging to the z/OS components deployed to that environment.

About this task

Files in the z/OS environment can be identified with the **Search Z Artifacts** option available in the z/OS application environment.

1. Navigate to the environment of interest (**Applications > *application name* > Environments > Environment: *environment name***.
2. From the menu options for the environment, select **Search Z Artifacts**.



3. Enter the name of the file to search in the **Search Z Artifacts** dialog, then click **Search.**

   You can refine your search result by using the following approaches:
   - Use wildcard syntax (*) after you enter a partial file or version name.
   - Limit the search scope by selecting any one of the applications, environments or components. The component filter is exclusive to an application or environment filter.

Notes:
- The search report displays the deployments done in z/OS agent v7.3.1 and later, and plug-in z/OS-Utility v80 and later.
- By default, the search result shows active records that match the searched criteria. You can include inactive records by selecting the **Show Inactive Records** checkbox.

Tips:
- File name search is case-insensitive and version name search is case-sensitive.
- The **File name** or **Version name** are required fields for searches.

The z/OS search report shows information in the columns below:

Table 762. z/OS Search report columns

Column	Description
Filename, **Inputs**, **Version Name**, and **Deploy Type**	Uses values from the version `shiplist.xml` file.
Component, **Version**, **Deployment Time**, **Application**, **Environment**, **Status**	Provides you deployment information based on the deployment target.

Column	Description
Target	Indicates the target PDS/PS or USS directory where the artifact is copied.
Target Type	Specifies the type of artifacts.
Operation Type	Shows the nature of action performed on the artifact in the target. For example, DEPLOY for created and updated artifacts, and DELETE for deleted artifacts.
(Download icon)	Downloads search results in the CSV format upon clicking this icon

Results

You have searched the files in z/OS environment inventory.

Related information

z/OS search on page 2548

Scheduling deployments

HCL DevOps Deploy (Deploy) has a built-in deployment scheduling system for setting regular deployments, or even black-out dates, for your Deployments.

Deployments for an individual application are scheduled on a per-environment basis, set when you run a deployment of a snapshot or deployment process. Black-out dates are set within the individual environments.

Deploy stores all times relative to Coordinated Universal Time and displays them based on your time zone. A process that you schedule during Daylight Saving Time moves back an hour after Daylight Saving Time ends.

To set up a scheduled deployment, click **Applications**, click your application, and next to the environment, click **Request Process**, and in the **Create Deployment** window, go to **Schedule Deployment**, and disable the **Run Now** option.

When you disable the **Run Now** option, the **Date** and **Time** fields appear for you to specify the date and time for the deployment to run. The **Recurrence Pattern** setting deploys the application on a regular schedule. For example, if you are practicing continuous delivery, the **Daily** option deploys the application to the target environment every day. You can choose the **Hourly**, **Daily**, **Weekly**, **Monthly**, and **Weekdays** recurrence pattern.

Note: When scheduling deployments for week days, if the deployment start date falls on a weekend, the deployment date gets scheduled for the following Monday at the defined start time.

After you schedule the deployment, it is added to the calendar. There, if you click the scheduled deployment, you can edit, delete, or investigate the deployment. Alternatively, click **Schedule Process** on the **Calendar** page to add scheduled deployments.

Notes:

- Do not change the starting time of a scheduled deployment. Instead, delete the recurring deployment and recreate it.
- Click **List**, on the **Calendar** page, to view the list of scheduled deployments for the chosen period. Click the enabled **List** to display the deployments in default view.

A *blackout* is a set per-environment, per-application. After the blackout is set, no deployments or snapshots can be scheduled to occur in that environment. Any previously scheduled deployments to the environment fail if they fall within the blackout date you set. To set up a blackout, go to (**Application > Environments > Calendar > Add Blackout**). If you must set blackouts for more than one environment, you must set a separate blackout for each environment. Deploy prompts you to give the dates and times for the blackout.

Calendar hotkey shortcuts

Table 763. Calendar hotkey shortcuts

Keyboard shortcut	Function
Double Click or Shift + Click	Brings up Schedule Deployment for the clicked date cell.
Ctrl + Click or Alt + Click inside an environment calendar	Enables you to set blackout date by dragging the mouse and releasing.

Canceling stalled deployments

There might be cases when the deployment processes fail to progress as expected. These stalled processes are listed in the deployment process log. You can cancel these stalled deployment processes.

Before you begin

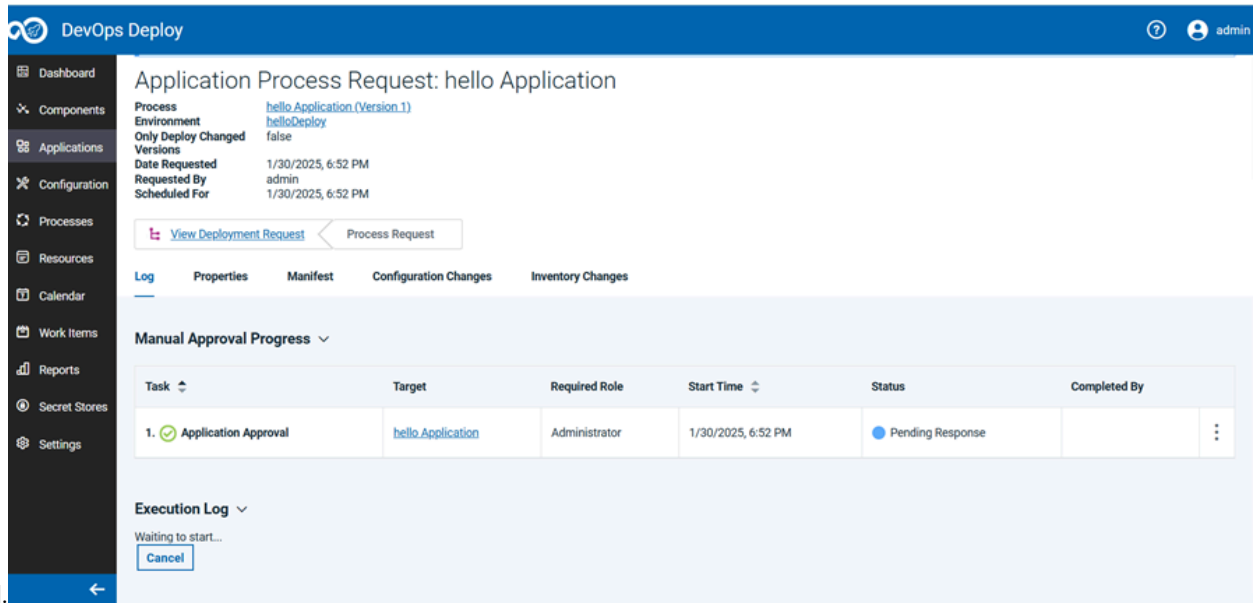
You must have permission to cancel stalled deployments.

About this task

You can cancel all the stalled deployment processes in the **Waiting to start** state.

1. Log in to Deploy.
2. Navigate to **Applications**.
   The list of applications is displayed.
3. Click the application for which you want to delete the stalled deployment processes.
4. Go to the **History** tab.
5. Click (**View Request**) for the process.

The **Log** tab is displayed.



6. Click **Cancel** in the **Execution Log** panel.

Results

The stalled deployment processes are canceled.

Reporting

HCL DevOps Deploy (Deploy)

Provides several different deployment and security reports.

Deployment reports contain historical information about deployments. Data can be filtered in various ways and reports can be printed and saved. In addition, you can save search criteria for later use. See Deployment reports on page 2547.

Security reports provide information about user roles and privileges. See Security reports on page 2549.

Deployment reports

Deployment reports contain historical information about deployments, such as the total number of deployment runs and the average duration of those runs.

To run a deployment report, select the filters at the top of the report, including the date range and application, and then click **Run**.

Deployment average duration

This report provides average deployment times for applications that ran during a specified reporting period. The report provides both a tabular presentation and line graph of the data.

Deployment total duration

This report provides total deployment times for applications that ran during a specified reporting period. The report provides both a tabular presentation and line graph of the data.

Deployment count

This report provides information about the number of deployments that ran during a specified reporting period. The report provides both a tabular presentation and a line graph of the data.

Deployment detail

This report provides a detailed, historical view of deployment data across applications, environments, and components. Each row in the report represents a deployment that ran during the reporting period and matches the filter conditions. Use this report to track application statuses, execution durations, utilized snapshots, request descriptions, scheduling timelines, and process approvers.

Note: To filter the deployment history, select one or more items from the **Application** list. If you select a single application, the **Environment** list becomes a multi-select drop-down menu of mapped environments. If you select multiple applications, or leave the **Application** list blank, the **Environment** list changes to a text search filter.

z/OS search

This search feature helps you search for a file name or version name in z/OS components in all environments and applications. Use this to search for a deployed artifact and find historical deployment information about the artifact. You can also search artifacts in rolled back versions using **Deploy-datasets**/ **rollback** step in the z/OS utility plug-in, which in its process updates the database inventory of the affected artifacts.

You can refine your search result by using the following approaches:

- Use wildcard syntax (*) after you enter a partial file or version name.
- Limit the search scope by selecting any one of the applications, environments or components. The component filter is exclusive to an application or environment filter.

Notes:

- The search report displays the deployments done in z/OS agent v7.3.1 and later, and plug-in z/OS-Utility v80 and later.
- By default, the search result shows active records that match the searched criteria. You can include inactive records by selecting the **Show Inactive Records** checkbox.

Tips:

- File name search is case-insensitive and version name search is case-sensitive.
- The **File name** or **Version name** are required fields for searches.

The z/OS search report shows information in the columns below:

Table 764. z/OS Search report columns

Column	Description
Filename, **Inputs**, **Version Name**, and **Deploy Type**	Uses values from the version `shiplist.xml` file.
Component, **Version**, **Deployment Time**, **Application**, **Environment**, **Status**	Provides you deployment information based on the deployment target.
Target	Indicates the target PDS/PS or USS directory where the artifact is copied.
Target Type	Specifies the type of artifacts.
Operation Type	Shows the nature of action performed on the artifact in the target. For example, DEPLOY for created and updated artifacts, and DELETE for deleted artifacts.
(Download icon)	Downloads search results in the CSV format upon clicking this icon

For information about saving and printing reports, see Saving and printing reports on page 2550.

Security reports

Security reports provide information about the user roles and privileges that are defined with the HCL DevOps Deploy (Deploy) security system.

You can sort each report by clicking on the column headers for things like the components and environments. Not all columns can be sorted.

Application security report

This report provides information about the user roles and privileges that are defined for applications in Deploy. Each report row represents an individual application.

Component security report

This report provides information about the user roles and privileges that are defined for components. Each report row represents an individual component.

Environment security report

This report provides information about the user roles and privileges that are defined for environments. Each report row represents an individual environment.

Resource security report

This report provides information about the user roles and privileges that are defined for resources. Each report row represents an individual resource.

User security report

This report provides information about the authentication realms and groups that are defined for users. Each report row represents an individual user. By default, the report runs automatically for all users. You can also filter the report based on authentication realms and groups.

For information about saving and printing reports, see Saving and printing reports on page 2550.

Setting up recurring reports

Recurring reports run on a schedule and e-mail the results to you.

Before you begin

- In the server system settings, in the **Mail Server Settings** section, configure an e-mail server.
- Make sure that your e-mail address in your account information is correct. Recurring reports are sent to that address.

About this task

Recurring reports can run monthly, weekly, or daily.

1. Create a report as usual.
2. Open the report.
3. Click **Schedule Recurring Report** .
4. Set the date on which to start the report, the time to run it, and the frequency for the report, and click **Submit**.
   You can choose the **Hourly**, **Daily**, **Weekly**, **Monthly**, and **Weekdays** recurrence pattern.

Results

The server runs the report on the schedule you set and e-mails a CSV file to your e-mail address.

What to do next

To cancel the recurring report, open the report, click **Schedule Recurring Report**, and select the **Stop scheduling this report?** check box.

Saving and printing reports

You can print and save report data for all report types. In addition, you can save filter and sort order information for deployment-type reports.

About this task

To save report data in a CSV (comma-separated value) file, follow these steps:

1. Set filters, if any, and run the report.
2. Click **Download CSV**.
3. From the browser dialog box, save the data to a file, or open the data by using an application that is associated with CSV files.

Note: The sort order and hidden or visible column information is not preserved in the CSV file.

To print a report, follow these steps:

1. Set the filter conditions.
2. Define the reporting period.
3. Run the report.
4. Optional: Set the sort order. Your changes are reflected in the printed report.
5. Optional: Change the column visibility. By default, all columns are selected to be shown in the printed report. Hidden columns are not shown in the output.
6. Click **Print** to print the report.

Saving report filters

You can save filter and sort-order settings for deployment reports. Saved reports can be retrieved with the **My Reports** menu on the **Reports** pane.

1. Set the filter conditions.
2. Define the reporting period.
3. Run the report.
4. **Optional:** Set the sort order.
   You can change the sort order for any column by clicking the column header.
5. **Optional:** Change column visibility. Click **Edit** to display the **Select Columns** dialog box. By default, all columns are selected to be shown in a report. To hide a column, select the corresponding check box.
6. Click **Save**.
7. Enter a name for the file, and save your work.

What to do next

To run your report, click the report name in the **My Reports** menu.

To delete your report, click **Delete**.

Monitor operating details with MBeans

With an external monitoring tool, such as IBM® Performance Management on Cloud 8.1.3 or New Relic, you can use Managed Beans (MBeans) to review numerous details about HCL DevOps Deploy (Deploy). For example, you can view statistics that specify how many components are available or see how long it takes for an application process to run to completion. You can use these statistics and details to assess the health of your deployments and deployment processes.

You can request and specify metrics by using four MBeans. Two MBeans provide summary information for all the targeted elements. For example, the agent summary information MBean provides aggregate information about all agents. The other two MBeans provide specific information about elements. The individual component information MBean provides details about a particular component, which you specify by name. The following list shows the four areas and the structure of the data that each MBean provides. Access the data according to the requirements of your monitoring tool. Refer to the documentation for the tool as required.

By default, the metrics monitoring is turned off. To enable the metrics monitoring, you must add a property to the `installed.properties` file. By default, this file is in the `/opt/install-dir/server/conf/server` folder. To enable monitoring, add this line to the file:

```
jmx.monitoring.enabled=true
```

If your servers are working in high availability, you can specify how frequently the servers synchronize data to ensure that the servers maintain a consistent list of components and applications. In the `installed.properties` file, add the following line:

```
jmx.monitoring.ha.update.interval.minutes=measurement_update_time
```

where *measurement_update_time* is a whole number is greater than or equal to 30. You cannot get updates more frequently than every 30 minutes. This limit reduces the processing requirements on the servers.

You can monitor four areas of operational details.

Agent summary information

JMX: com.urbancode.ds.agents:type=monitor

- `AllAgentCount` (The total number of agents)
- `ConnectingAgentCount` (The number of agents that are currently attempting to connect to the server)
- `UpgradeRecommendAgentCount` (The number of agents to upgrade to achieve best results)
- `UpgradeRequiredAgentCount` (The number of agents that must be upgraded)
- `ErrorAgentCount` (The number of agents that are in an error state)
- `OfflineAgentCount` (The number of agents that are offline)
- `InstallableAgentCount` (The number of agents that can be installed)
- `OnlineAgentCount` (The number of agents that are online)
- `LicensedAgentCount` (The number of the agents that have licenses)
- `UnlicensedAgentCount` (The number of unlicensed agents)

Individual component information

JMX: com.urbancode.ds.components.individual.monitor:type=

COMPONENT_NAME

`LatestVersionSizeInBytes` (The size in bytes of the most recent version of the specified component)

`PreviousVersionSizeInBytes` (The size in bytes of the previous version of the specified component)

`AverageVersionSizeInBytes` (The average in bytes size of all the versions of the specified components)

You can specify multiple components individually in three ways:

- Name the component specifically, for example:

```
com.urbancode.ds.components.individual.monitor:type=
     ComponentA

com.urbancode.ds.components.individual.monitor:type=
     ComponentB
```

- Use wildcards to report components with names that match the sample, for example:

```
com.urbancode.ds.components.individual.monitor:type=
     Component*
```

- Use a wildcard to report all components, for example:

```
com.urbancode.ds.components.individual.monitor:type=*
```

Component summary information

JMX: com.urbancode.ds.components.summary:type=monitor

`TotalComponentCount` (The total number of components)

`FailedImportComponentCount` (The number of components whose latest version was not imported after an import request)

`ImportingComponentCount` (The number of components that are currently being imported)

`NoVersionComponentCount` (The number of components that do not have versions)

`OKComponentCount` (The number of components whose latest versions were not imported successfully)

Individual application information

JMX:com.urbancode.ds.applications.individual.monitor:type=

APPLICATION_NAME

`LatestDeploymentDurationInMS` (The amount of time that the most recently completed deployment process lasted for the specified component)

`PreviousDeploymentDurationInMS` (The amount of time that the previously completed deployment process lasted for the specified component)

`AverageDeploymentDurationInMS` (The average amount of time that completed deployment processes lasted for the specified component)

- Name the component specifically, for example:

```
com.urbancode.ds.components.individual.monitor:type=
      ApplicationA

com.urbancode.ds.components.individual.monitor:type=
      ApplicationB
```

- Use wildcards to report components with names that match the sample, for example:

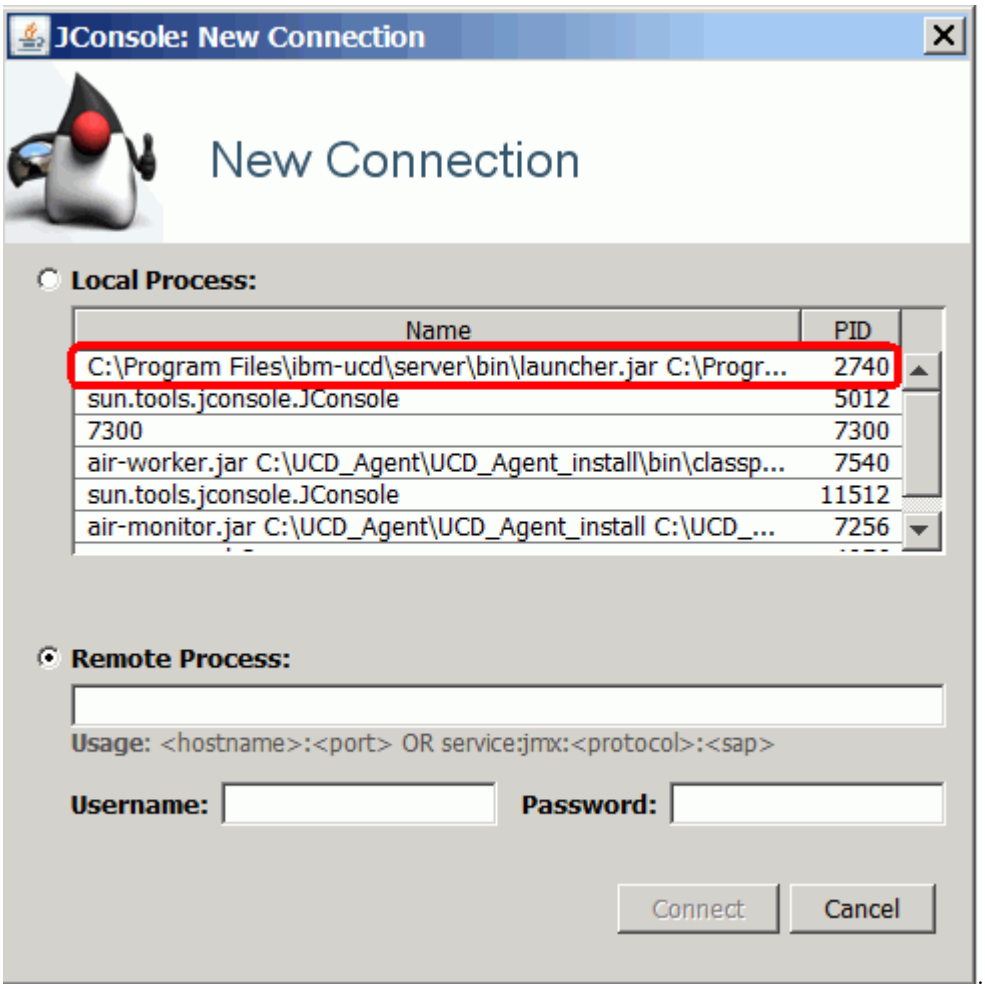
```
com.urbancode.ds.components.individual.monitor:type=
      Application*
```

- Use a wildcard to report all components, for example:

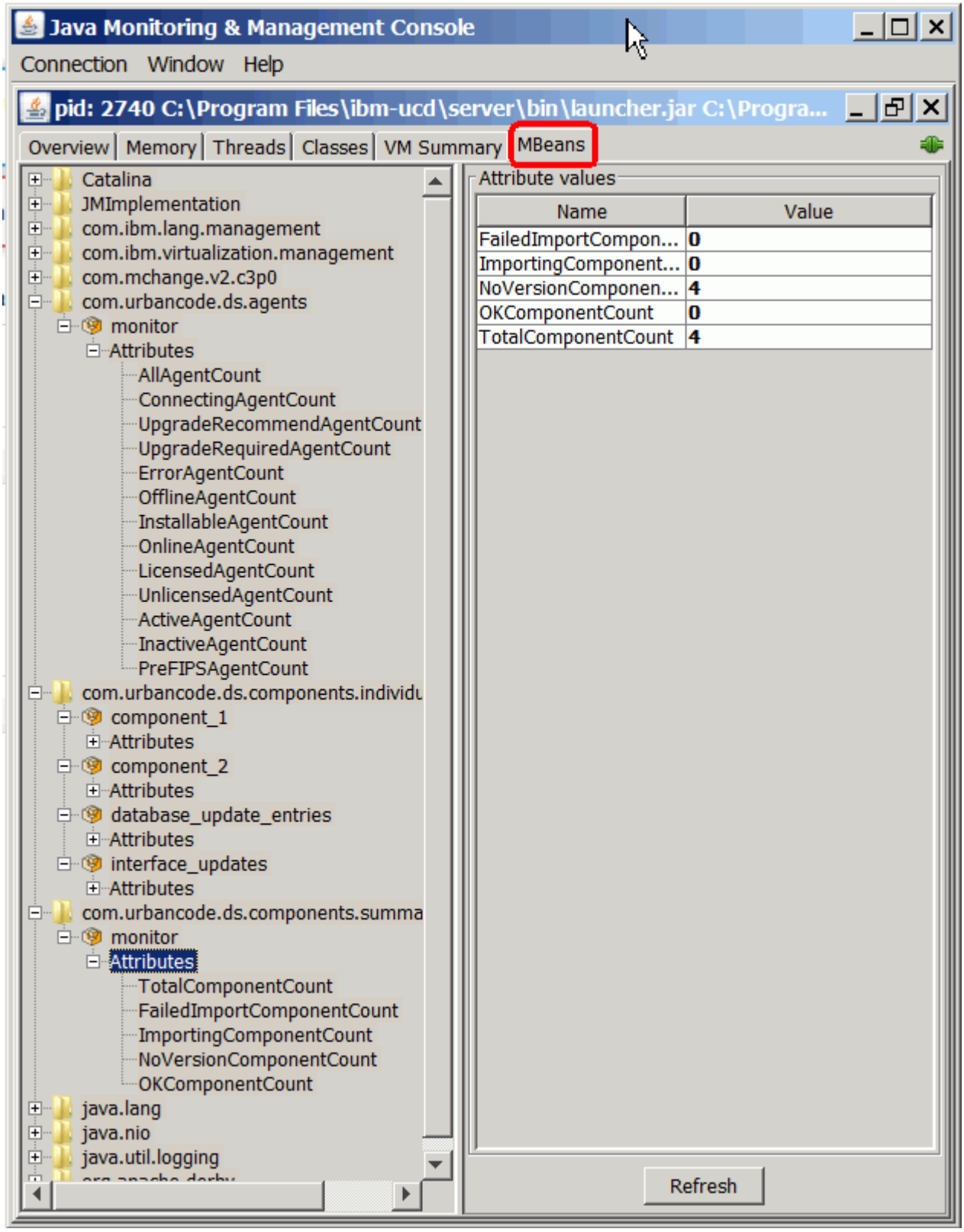
```
com.urbancode.ds.components.individual.monitor:type=*
```

Viewing metrics with JConsole

JConsole is one external monitoring tool that is immediately available to you. For information about JConsole and starting the tool, see Using JConsole. After you turn on monitoring as described earlier and start JConsole, select a new connection as shown in this illustration:



Then, click the **Mbeans** tab. In the left navigation, select the metrics to view.



Elasticsearch and Kibana reports

Use your project data to create Elasticsearch reports and Kibana charts.

If you use Elasticsearch and Kibana for data analytics, you can use HCL DevOps Deploy (Deploy) data with your ELK stack environment. The following reports are currently available

API requests

This report provides data about API requests made to the Deploy server.

Application process requests

This report contains application process requests such as average duration, and average warning count.

Codestation

This report contains data about Codestation usage including total size and cleanup counts.

Information about configuring your ELK stack for Deploy is found in this post.

Extending Product Function

Learn how to extend the function of HCL DevOps Deploy (Deploy).

Plug-ins

The Deploy plug-ins provide tools for creating component processes and integrations. HCL provides plug-ins for several common deployment processes, and others are available to integrate with a wide variety of tools, such as middleware tools, databases, and servers.

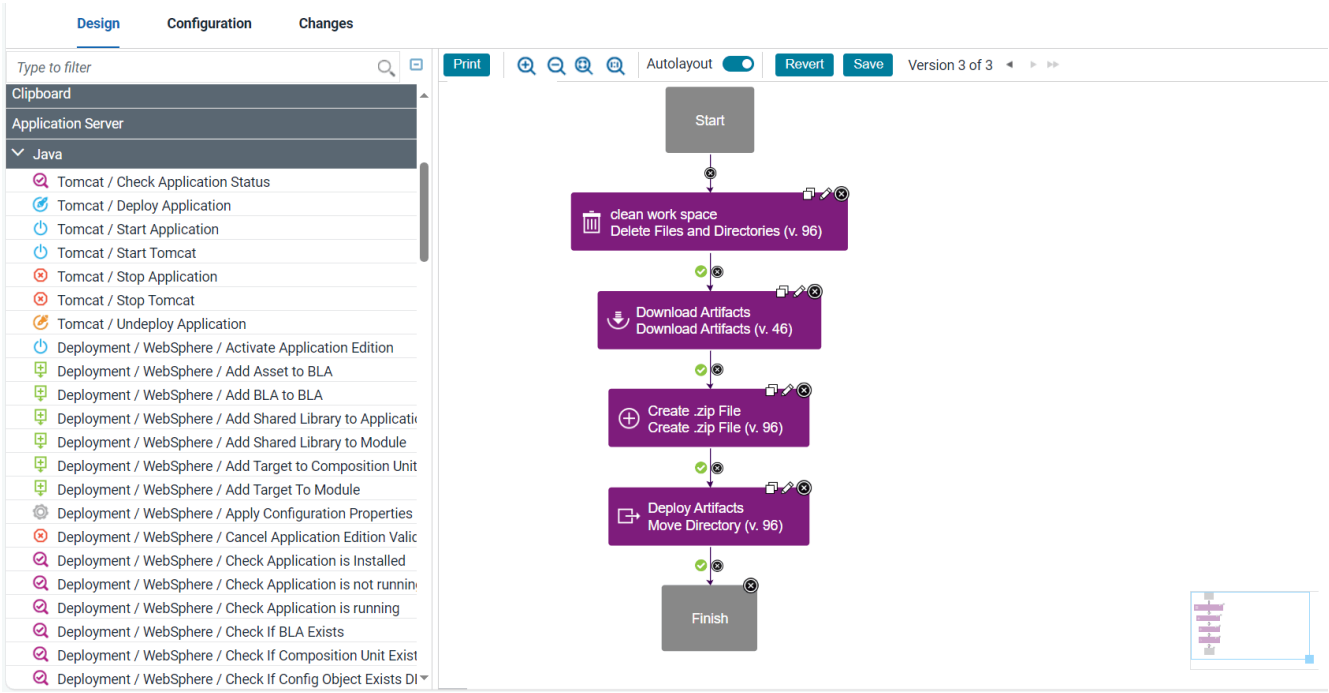
Plug-ins provide processing and integration functions. There are two types of plug-ins:

- *Source-type* plug-ins integrate with external systems to import artifacts and create component versions.
- *Automation-type* plug-ins provide process steps that manipulate components, typically by deploying them.

Both types of plug-ins consist of distinct processes called *steps*. Each step consists of a number of properties, a command that runs the step, and post-processing instructions. (Post-processing typically ensures that expected results occur.) Step properties can serve various purposes, from providing input to the command to supplying some, or all, the actual command itself. Automation-type plug-in properties can be set at design time in the process editor, or at run time in the user interface. Source-type plug-in properties are typically defined when a component is created.

Automation-type plug-ins

You use automation plug-ins to deploy components or otherwise manage them. Each step is a self-contained unit with inputs, outputs, and properties. By combining automation plug-in steps in the process editor, you can create fully automated deployment processes. When you create a process, you drag steps onto the design area and define the properties as you go. Property values can be specified when you define the process or at run time. The process flow is defined by drawing connections between steps. The following illustration, shows a series of automation-type plug-in steps and the connections between them.



For information about creating component processes, see Component processes on page 2299.

At deployment time, component processes are run by agents that are installed in the target environment. For a process to run, the agent must have access to all resources, tools, and files that are required by the plug-in steps in the process. When you install an agent, ensure that these conditions are met:

- The agent that runs the process has the necessary user permissions to run commands and can access required resources. Assigning these permissions typically entails granting permissions if an external tool is installed as a different user; installing the agent as a service; or impersonating the appropriate user. See User impersonation for process steps on page 2462.
- External tools that plug-in steps require are installed in the target environment.
- The required minimum version of an external tool is installed.

For information about installing agents, see Installing agents from the command line.

Source-type plug-ins

You use source plug-ins to import artifacts and create component versions. A source plug-in defines the type of artifacts that can be imported and identifies where the artifacts are located. Source-type plug-ins are configured when components are created, you do not use the process editor to configure source-type plug-ins. Unlike automation plug-ins, source plug-ins always have a single step, the Import Version step.

Artifacts are imported into component versions by agents that are running in the environment where the artifacts are located. For a component version to be created, the agent must have the required permissions and access to all required resources and files. For information about installing agents that run source plug-ins, see System settings on page 2086.

For information on how to use the source config plug-ins that are provided with Deploy, see Creating components on page 2248.

Related information

Installing plug-ins from the Web UI on page 2560

Installing plug-ins from the plug-in site on page 2559

Installation of plug-ins

You can install plug-ins from the Deploy Web UI and the plug-ins site.

Deploy plug-ins are categorized into two types; core plug-ins and external plug-ins.

Core plug-ins

Plug-ins that are shipped and pre-installed in Deploy are called the core plug-ins. In an initial installation, the core plug-ins are listed on the **Settings > Automation > Installed Plugins** page. Alternatively, these plug-ins are available to download and their documentation is also available from the HCL plug-ins site.

The following table lists the core plug-ins:

Plug-in	Plug-in Type	Description
ClearCase	Source Configuration	Automates importing version artifacts from ClearCase (Base and UCM).
DevOps Build	Source Configuration	Automates uploading artifacts from a DevOps Build server.
File System	Source Configuration	Automates uploading artifacts from a normal file system.
File System (Versioned)	Source Configuration	Automates uploading artifacts from a versioned file system.
Git	Source Configuration	Automates importing artifacts from Git.
Maven	Source Configuration	Automates downloading artifacts from Maven projects.
Subversion	Source Configuration	Automates importing artifacts from a Subversion repository.
TFS_SCM	Source Configuration	Automates importing version artifacts from a TFS server.
zOS File	Source Configuration	Automates uploading data sets, UNIX files, and generic artifacts from a z/OS system.
File Utils	Automation	Includes steps, such as deleting or creating directories and replacing tokens in a file, that perform folder-level and file-level tasks as part of a deployment process.
General Utilities	Automation	Provides utilities for basic tasks that don't fit in a specific plug-in.
Groovy	Automation	Provides steps for executing user-defined Groovy scripts.

Plug-in	Plug-in Type	Description
Kubernetes	Automation	Includes steps to work with resources and images in Kubernetes.
OpenShift	Automation	Includes steps to work with resources and images in Red Hat OpenShift.
Shell	Automation	Allows users to run custom shell scripts during the deployment process.
System Information	Automation	Includes a variety of checks to perform against the operating system. These steps can be used to verify that a deployment can succeed or has succeeded.
Launch Applications	Automation	Creates and manages Deploy applications.
Launch Component	Automation	Creates and manages Deploy components.
Launch Configuration Management	Automation	Downloads and uploads configuration templates from/to Deploy.
Launch Environments	Automation	Creates and manages Deploy environments.
Launch Processes	Automation	Supports interaction with processes and process requests.
Launch Resources	Automation	Manages Deploy resources.
Launch Versioned File Storage	Automation	Allows Deploy to upload artifacts to CodeStation.
Launch Versions	Automation	Allows editing Component Versions in Deploy.
zOS Multi Generate Artifact Information	Automation	Generates one or more informational texts based on defined templates.
zOS Utility	Automation	Includes steps related to deployment activities for z/OS.

External plug-ins

External plug-ins are the plug-ins that you must install to be able to use them. Deploy supports integration with many external applications. You can install the external plug-ins by using the following methods:

- Installing plug-ins from the plug-in site on page 2559
- Installing plug-ins from the Web UI on page 2560

Note: The plug-ins are developed and updated independently of the Deploy release cycles.

Installing plug-ins from the plug-in site

You can install plug-ins by downloading them from the plug-ins site and loading them into the Deploy product.

About this task

You can browse, download, and get documentation for external plug-ins from the HCL Plug-ins site. The site contains many plug-ins that are available in the Deploy product.

The site includes plug-ins that are developed by the following sources:

- **HCL Software** — Your Deploy license entitles you to download and install the plug-ins that are developed by HCLSoftware.
- **Partners** — Plug-ins that are provided by partners can be purchased from the partner website. Search for `partner` to find the partner plug-ins in the plug-ins site.

  Note: Support for each partner plug-in is provided exclusively by the partner who provided the plug-in.

- **Community** — Plug-ins are provided by members of the Deploy community on the HCLSoftware Deploy Community website. Search for `community` to find the community plug-ins in the plug-ins site. You can contribute to code in the community or use code posted there to incorporate into your own.

  Note: Support is provided exclusively through the community.

1. On the Web UI, go to **Settings > Automation > Installed Plugins**.
2. Load the plug-in by clicking **Load Plugin**.
3. Enter the path to the compressed plug-in file, and then click **Submit**.

Results

The plug-in is installed and listed on the **Installed Plugins** page. After the plug-in is installed, it is available immediately.

Notes:

- You do not need to restart the server after you install a plug-in.
- You can upgrade a plug-in by installing the new version over the old version.
- You cannot roll back a plug-in version to a previous version from the Web UI. To move to a previous version of a plug-in, see Rolling back a plug-in version to a previous version.

Installing plug-ins from the Web UI

To avoid navigating to the plug-ins site, you can seamlessly install the latest plug-in versions directly from the Web UI.

Before you begin

You must have ensured that the server has network access and can reach the plug-ins site.

Note: If the server cannot access the site because the site location has changed since your product version was released, then you must set the `com.urbancode.ds.web.cli.rest.plugin.PluginsManifest.url` property in the

`appdata/conf/server/getting-started-custom-urls.properties` file to a reachable URL for the plug-ins. For example, you must set the property as:

```
com.urbancode.ds.web.cli.rest.plugin.PluginsManifest.url=https://
ucd-plugins.example.com/path/to/manifest.json
```

1. Go to the **Settings > Automation > Available Plugins**.

   Note: To view information about a plugin, click **View Documentation** from the **Actions** menu. This option opens the documentation site for that specific plugin.

2. Search for the plug-in you want to install.
3. Click the menu option on the right of the plug-in for which you want to install.
4. Click **Install**.

Results

The plug-in is installed and listed on the **Installed Plugins** page. After the plug-in is installed, it is available immediately.

Notes:

- You do not need to restart the server after you install a plug-in.
- You can upgrade a plug-in by installing the new version over the old version.
- You cannot roll back a plug-in version to a previous version from the Web UI. To move to a previous version of a plug-in, see Rolling back a plug-in version to a previous version.

Creating plug-ins

You can create your own plug-ins for use with HCL DevOps Deploy (Deploy). A plug-in consists of XML files and supporting script files that the plug-in requires.

- The `plugin.xml` file defines the steps that constitute the plug-in; plug-in steps define a plug-in's functions. Each step is an independently configurable entity in the Deploy editor.
- The `info.xml` file contains a version ID and other information that describes the plug-in.

Note: The plug-in files must use UTF-8 encoding.

A plug-in step is defined by a `<step-type>` element that contains these elements:

- One `<properties>` element
- One `<command>` element
- One `<post-processing>` element

The `<properties>` element is a container for `<property>` child elements, and can contain any number of `<property>` elements. Property values can be specified at design time or run time. The `<post-processing>` element provides error-handling capabilities and sets property values that other steps can use. The `<command>` element runs the step's function. The function can be defined completely by the element or be constructed in part or entirely from the step's properties at design time or run time.

A command can access these properties:

- The step's own properties
- Properties that were set earlier by other steps in the process
- Properties that were set by the application that started the process
- Properties on the target environment and resource

Step property values become unavailable after the component process ends.

Plug-in steps are run by an agent that is installed in the target environment. Thus, plug-ins can be written in any scripting language, and the agent can access the required scripting tools on the host. After a plug-in is created, upload it into Deploy to make it available to users. To upload a plug-in, create an archive file that contains the XML files (`plugin.xml` and `info.xml`) and scripts that the plug-in requires. Then, import the archive file with the **Automation Plugins** pane for automation-type plug-ins or the **Source Config Plugins** pane for source plug-ins.

To troubleshoot plug-ins, you can retrieve the information from each step that ran in a process request. To download these logs, open the process request and click **Download All Logs**.

The `plugin.xml` file for automation plug-ins

An automation plug-in is defined with the `plugin.xml` file.

The structure of this file consists of a `header` element and one or more `step-type` elements. The `header` element identifies the plug-in. Each `step-type` element defines a step. Steps are available to users in the process editor and are used to construct component processes.

After the document type declaration, the `plugin` root element identifies the XML schema type, PluginXMLSchema_v1.xsd, which all plug-ins use. The following code sample presents the basic structure of the `plugin.xml` file for automation plug-ins:

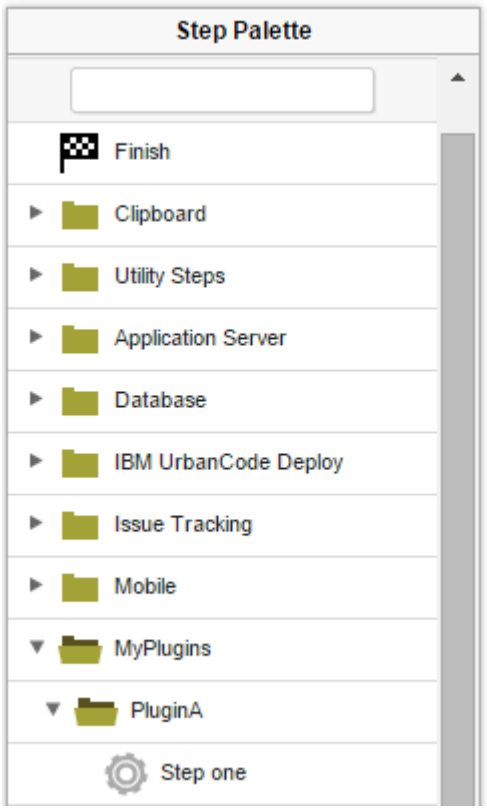
```
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://www.urbancode.com/PluginXMLSchema_v1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <header>
    <identifier id="plugin_id" version="version_number" name="Plug-in Name"/>
    <description/>
    <tag>Plugin type/Plugin subtype/Plugin name</tag>
  </header>
  <step-type name="Step_Name">
    <description/>
    <properties>
      <property name="property_name" required="true">
         <property-ui type="textBox" label="Driver Jar"
```

```
            description="The full path to the jdbc driver jar to use."
            default-value="${p:resource/sqlJdbc/jdbcJar}"/>
      </property>
    </properties>
    <post-processing>
      <![CDATA[
        if (properties.get("exitCode") != 0) {
            properties.put("Status", "Failure");
        }
        else {
            properties.put("Status", "Success");
        }
     ]]>
    </post-processing>

    <command program="${path_to_tool">
      <arg value="parameters_passed_to_tool"/>
      <arg path="${p:jdbcJar}"/>
      <arg file="command_to_run"/>
      <arg file="${PLUGIN_INPUT_PROPS}"/>
      <arg file="${PLUGIN_OUTPUT_PROPS}"/>
    </command>
  </step-type>
</plugin>
```

The `<tag>` tag specifies the location of the steps in the process editor palette. This is a required tag and the tag value cannot be null. For example, the following code specifies that the steps are in the `MyPlugins/PluginA` folder.

```
<tag>MyPlugins/PluginA</tag>
```

When the plug-in is installed, the steps appear in the specified folder in the palette, as shown in the following figure.



You can further organize specific steps into sub-folders. For example, assume that the `plugin.xml` file puts steps in the `MyPlugins/PluginA` folder, as in the previous example. The following code puts a step in the `MyPlugins/PluginA/Subgroup1` sub-folder:

```
<step-type name="Step one">
  <server:sub-tag>Subgroup1</server:sub-tag>
```

To organize steps into sub-folders in this way, you must add the namespace `xmlns:server="http://www.urbancode.com/PluginServerXMLSchema_v1"` to the `<plugin>` tag that is at the root of the `plugin.xml` file, as in the following example:

```
<plugin xmlns="http://www.urbancode.com/PluginXMLSchema_v1"
  xmlns:server="http://www.urbancode.com/PluginServerXMLSchema_v1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

For information about the structure of `plugin.xml` for source plug-ins, see Creating source plug-ins on page 2566.

The `header` element

The mandatory `header` element identifies the plug-in and contains three mandatory child elements and one optional child element:

Table 765. Contents of the `header` element

`<header>` Child Elements	Description
`<identifier>`	This element's three attributes identify the plug-in:

Table 765. Contents of the `header` element (continued)

`<header>` Child Elements	Description
	• `version` API version (the version number for upgrading plug-ins is defined in the `info.xml` file). • `id` Identifies the plug-in. • `name` The plug-in name is shown on the **Automation Plugins** pane, and on informational pages for the plug-in. All values must be enclosed within single quotation marks.
`<description>`	Describes the plug-in; is shown on the **Automation Plugins** pane, and on informational pages for the plug-in.
`<tag>`	Defines where the plug-in is listed within the process editor's hierarchy of available plug-ins. The location is defined by a string that is separated by slashes. For example, the Tomcat definition is the following string: `Application Server/Java/Tomcat`. The Tomcat steps are listed beneath the Tomcat item, which in turn is nested within the other two. **Note:** Do not include underscores in this element.
`<server:required-server-version>`	This optional element defines the minimum version of HCL DevOps Deploy (Deploy) required for use with the plug-in. The version is defined by 4 digits. For example, a plug-in that requires Deploy Version 7.1.0 or later would include the following element in the header: `<server:required-server-version>7.1.0.0</server:required-server-version>`
`<server:dependencies>`	This optional element specifies other plug-ins that must be loaded before this plug-in. The `<server:dependencies>` element contains a `<server:dependency>` element that specifies the identifier and minimum version for each dependency. For example, a plug-in that requires version 76 or later of the WebSphere® Application Server - Deploy plug-in would include the following elements in the header:

Table 765. Contents of the `header` element (continued)

`<header>` Child Elements	Description

	`<server:dependencies>`
	`  <server:dependency plugin-id="com.urbancode.air.plugin.WebSphere"`
	` min-version="76"/>`
	`</server:dependencies>`

The following text is a sample header definition:

```
<header>
  <identifier version="3" id="com.&company;.air.plugin.Tomcat" name="Tomcat"/>
  <description>
  The Tomcat plugin is used during deployments to run Tomcat run-book
  automations and deploy or undeploy Tomcat applications.
  </description>
  <tag>Application Server/Java/Tomcat</tag>
</header>
```

The `plugin-javaopts.conf` file

The `plugin-javaopts.conf` file can be used to set the *JAVA_OPTS* environment variable for automation plug-ins that use Groovy.

By default, the `plugin-javaopts.conf` file is stored on the agent computer in the *`agent_installation_directory`*`/conf/` folder. You can use the `plugin-javaopts.conf` file to set the *JAVA_OPTS* environment variable for a plug-in that is different from the *JAVA_OPTS* variable for the agent process itself. The plug-in must use Groovy, and not Java™.

Note: If there are syntax errors in the `plugin-javaopts.conf` file, errors and unpredictable behavior can result. Also, any custom modifications that you make to the `plugin-javaopts.conf` file might be overwritten if you upgrade the agent to a newer version.

JVM options for optimizing performance

When you run a plug-in step, the agent automatically adds the appropriate Java optimization options to the `plugin-javaopts.conf` file, if it's not available. For example, if you use the IBM Java Runtime Environment (JRE), the agent adds the `-Xquickstart` parameter automatically to the `plugin-javaopts.conf` file.

If you use Oracle JDK or Open JDK distributions, the agent adds the `-XX:TieredStopAtLevel=1` parameter to the `plugin-javaopts.conf` file to optimize the plug-in step run time.

Creating source plug-ins

Source-type plug-ins are defined the same way that automation-type plug-ins are defined except for some modifications to the `plugin.xml` file.

Like automation plug-ins, source plug-ins are defined with two files, an `info.xml` file and a `plugin.xml` file. The structure of the first file is identical for both plug-in types. The structure of the `plugin.xml` file varies depending on the type of plug-

in. For information about the `plugin.xml` file for automation plug-ins, see The plugin.xml file for automation plug-ins on page 2562.

The following code fragment is an example of a source plug-in file. The `<server:plugin-type>Source</server:plugin-type>` tag identifies the type of plug-in. For source plug-ins, the type must be `Source`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- the xmlns:server argument is necessary to parse the tags with the server prefix -->
<plugin xmlns="http://www.urbancode.com/PluginXMLSchema_v1"
    xmlns:server="http://www.urbancode.com/PluginServerXMLSchema_v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <header>
    <identifier id="com.urbancode.air.plugin.source.Name" name="Plugin Name" version="1"/>
    <description>Short description</description>
    <tag>SCM/Name</tag>

    <server:plugin-type>Source</server:plugin-type>
  </header>
```

A source plug-in has one step, which must be named `Import Version`:

```
    <step-type name="Import Version">
     <description>Creates a new component version and imports artifacts</description>
     <properties>
     </properties>
     <post-processing><![CDATA[
       if (properties.get("exitCode") != 0) {
           properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
       }
       else {
           properties.put("Status", "Success");
       }
    ]]></post-processing>

     <command program="${JAVA_HOME}/bin/java">
       <arg value="${JAVA_OPTS}"/>
       <arg value="-jar"/>
       <arg file="FileSystemSourceConfig.jar"/>
       <arg value="ImportVersion"/>
       <arg file="${PLUGIN_INPUT_PROPS}"/>
       <arg file="${PLUGIN_OUTPUT_PROPS}"/>
     </command>

   </step-type>
```

As shown in the following code fragment, source plug-ins have two property groups. `Component` type properties are configured with the **Create Component** window. `Import` type properties are configured with the **Import Component** window.

```
    <!-- ==================================== -->
    <!--   Properties                         -->
    <!-- ==================================== -->

    <server:property-group type="Component" name="FileSystemComponentProperties">
      <server:property name="property-name" required="true">
        <server:property-ui type="textBox"
                            label="property-label"
                            description="property-description" />
      </server:property>
    </server:property-group>

       <!-- If left empty or omitted then no dialog will appear -->
    <server:property-group type="Import" name="FileSystemImportProperties">
      <server:property name="property-name" required="true">
        <server:property-ui type="textBox"
                            label="property-label"
                            description="property-description" />
      </server:property>
    </server:property-group>
</plugin>
```

Plug-in steps: The `step-type` element

You define plug-in steps with the `step-type` element; each `step-type` element represents a single step in the HCL DevOps Deploy (Deploy) process editor.

A `step-type` element has a name attribute and up to four child elements:

- `description`
- `properties`
- `command`
- `post-processing`

The mandatory name attribute identifies the step. The description and name are shown in the web application and on informational pages for the plug-in. The contents of the description determine which icon is displayed with the step.

```
<step-type name="Start">
<description>Start Apache HTTP server</description>
```

Note: A step name cannot contain the forward slash (/) character.

Step properties: the `properties` element

The `properties` element is a container for properties, which are defined with the `property` tag.

Each step has a single `properties` element; a `properties` element can contain any number of `property` child elements.

A `property` tag has a mandatory `name` attribute, optional `required` attribute, and two child elements, `property-ui` and `value`, which are defined in the following table.

Table 766. The `<property>` element

`<property>` Child Elements	Description
`<property-ui>`	Defines how the property is presented to users in the HCL DevOps Deploy (Deploy) editor. This element has several attributes: • `label` Identifies the property in the editor's **Edit Properties** dialog box. • `description` Text that is displayed to users in the associated roll-over help box. • `default-value` Property value that is displayed when the **Edit Properties** dialog box is displayed; used if unchanged. • `type` Identifies the type of widget that is displayed to users. Possible values are: ◦ `textBox` Enables users to enter an arbitrary amount of text, limited to 4064 characters. ◦ `textAreaBox` Enables users to enter an arbitrary amount of text (larger input area than textBox), limited to limited to 4064 characters. ◦ `secureBox` Used for passwords. Similar to textBox except values are redacted. ◦ `checkBox` Displays a check box. If checked, a value of true is used; otherwise the property is not set. ◦ `selectBox` Requires a list of one or more values, which are displayed in a list. Configuring a value is described below.

Table 766. The `<property>` element (continued)

`<property>` Child Elements	Description
`<value>`	Used to specify values for a selectBox. Each value has a mandatory `label` attribute, which is displayed to users, and a value that is used by the property when selected. Values are displayed in the order they are defined.

Here is a sample `<property>` definition:

```
<property name="onerror" required="true">
  <property-ui type="selectBox"
    default-value="abort"
    description="Action to perform when statement fails: continue, stop, abort."
    label="Error Handling"/>
  <value label="Abort">abort</value>
  <value label="Continue">continue</value>
  <value label="Stop">stop</value>
</property>
```

Plug-in step icons

A keyword in the description determines which icon is displayed in the plug-in step.

In most cases, the first word in the description of the plug-in step determines the icon that is displayed. The following table shows the available icons, the keywords that are associated with the icons, and whether the keyword must be the first word in the description.

Table 767. Plug-in step icons

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Activate	Yes	start	
Add	Yes	add	
Allocate	Yes	allocate	
AppCmd	Yes	shellArgs	
Approval	No	setStatus	

Table 767. Plug-in step icons (continued)

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Assign	Yes	set	
Associate	Yes	link	
Batch Import	Yes	import	
Cancel	Yes	stop	
Change	Yes	modify	
Check	Yes	check	
Configure	Yes	configure	
Copy	Yes	copy	
Delete	Yes	delete	
Deploy	No	deploy	
Define	No	modify	
DesiredInventoryAction	No	process	
Disable	Yes	stop	
Download	No	download	

Table 767. Plug-in step icons (continued)

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Each	No	runProcess	
Enable	Yes	start	
Execute	Yes	start	
Export	Yes	export	
Flip	Yes	flip	
FTP	Yes	ftp	
Get	Yes	read	
Import	Yes	import	
Install	No	install	
Launch	Yes	deploy	
Link	Yes	link	
Modify	Yes	modify	
Monitor	Yes	monitor	
Move	Yes	move	

Table 767. Plug-in step icons (continued)

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Manual Task	No	manualTask	
Process	Yes	runProcess	
Publish	Yes	publish	
Read	Yes	read	
Reboot	Yes	restart	
Recycle	Yes	recycle	
Register	Yes	create	
Receive	Yes	receive	
Retrieve	Yes	receive	
Restart	Yes	restart	
Revert	Yes	revert	
Rollback	No	revert	
Run	No	runProcess	
Run Groovy Script	Yes	runScript	

Table 767. Plug-in step icons (continued)

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Send	No	send	
Shell	Yes	shell	
Shell (with xargs)	Yes	shellArgs	
Start	Yes	start	
Stop	Yes	stop	
Set	Yes	set	
Submit	Yes	setStatus	
Sync	Yes	sync	
Terminate	Yes	stop	
Unzip	Yes	unzip	
Untar Tarball	Yes	unzip	
Update	Yes	sync	
Upload	Yes	upload	
Upgrade	Yes	upgrade	

Table 767. Plug-in step icons (continued)

Step keyword	Keyword must be the first word in the description	Image name	Displayed image
Undeploy	Yes	undeploy	
Uninstall	No	uninstall	
Untar	No	unzip	
Validate	Yes	setStatus	
Verify	Yes	check	
View	Yes	monitor	
Wait	Yes	wait	
WebSphere®	No	websphere	

The `command` element

Steps are run by starting the scripting tool or interpreter that is specified by the `<command>` element. The `<command>` element's `program` attribute defines the location of the tool that runs the command.

It bears repeating that the tool must be on the host and the agent that starts the tool must have access to it. In the following example, the location of the tool that runs the command (the Java™ scripting tool Groovy in this instance) is defined.

```
<command program='${GROOVY_HOME}/bin/groovy'>
```

The actual command and any parameters it requires are passed to the tool by the `<command>` element's `<arg>` child element. Any number of `<arg>` elements can be used. The `<arg>` element has several attributes:

Table 768. `<arg>` element attributes

Attribute	Description
`<value>`	Specifies a parameter that is passed to the tool. Format is tool-specific; must be enclosed by quotation marks.
`<path>`	Path to files or classes that are required by the tool. Must be enclosed by quotation marks.
`<file>`	Specifies the path to any files or classes that are required by the tool. Format is tool-specific; must be enclosed by quotation marks.

Because `<arg>` elements are processed in the order they are defined, ensured the order conforms to the order that is expected by the tool.

```
<command program='${GROOVY_HOME}/bin/groovy'>
  <arg value='-cp' />
  <arg path='classes:${sdkJar}:lib/commons-codec.jar:
    lib/activation-1.1.1.jar:
    lib/commons-logging.jar:lib/httpclient-cache.jar:
    lib/httpclient.jar:lib/httpcore.jar:
    lib/httpmime.jar:lib/javamail-1.4.1.jar' />
  <arg file='registerInstancesWithLB.groovy' />
  <arg file='${PLUGIN_INPUT_PROPS}' />
  <arg file='${PLUGIN_OUTPUT_PROPS}' />
</command>
```

The `<arg file='${PLUGIN_INPUT_PROPS}'/>` specifies the location of the tool-supplied properties file. The `<arg file='${PLUGIN_OUTPUT_PROPS}'/>` specifies the location of the output file for the step-generated properties.

Note: new lines are not supported by the `<arg>` element and are shown in this example only for presentation.

The `post-processing` element

When a plug-in step's `<command>` element finishes processing, the step's mandatory `<post-processing>` element runs.

The `<post-processing>` element sets the step's output properties (step name/property name, see Output properties on page 2516) and provides error handling. The `<post-processing>` element can contain any valid JavaScript™ 1.7 script (unlike the `<command>` element, `<post-processing>` scripts must be written in JavaScript™ 1.7). Users can also provide their own scripts when they define the step in the editor; see Post-processing scripts on page 2415.

Post-processing scripts have access to the following objects:

- The **properties** object is an instance of the class java.util.Properties. This variable has several special properties: `exitCode` contains the process exit code and `Status` contains the step's status. A `Status` value of `Success` means that the step completed successfully.

  This object is the same Java™ properties object that the step returns. Therefore, you can use the **properties** object in the post-processing script to change the output properties of the step. For example, to add an output property that is named `OutputProp1` with the value `value1`, add the following code to the post-processing script:

  ```
  properties.put("OutputProp1","value1");
  ```

  Then, the precondition scripts of other steps can access this property. See Process step preconditions on page 2411.

  For more information about this object, see the Java reference for java.util.Properties.

- The **scanner** object can scan the step's output log (scanning occurs on the agent) and run commands based on the results. **scanner** has several public methods:
  - `register(String regex, Function script)` registers a function to be called when the regular expression is matched. The function accepts two parameters: the first parameter is the line number that matched the regular expression, and the second parameter is the line of text that matched that expression.
  - `addLOI(Integer lineNumber)` adds a line to the lines of interest list, which are highlighted in the Log Viewer. The scanner adds all matching lines to this list automatically, but you can add other lines by calling this method.
  - `getLinesOfInterest()` returns a java.util.List of lines of interest; can be used to remove lines.
  - `scan()` scans the log. Call this method after you have registered all regular expressions with register(). If you do not call this method, the registered matchers do not run.
- The **commandOut** object is an instance of the class java.io.PrintStream. You can use this object to write to the log, as in the following example:

  ```
  commandOut.println("Log message.");
  ```

  For more information about this object, see the Java reference for java.io.PrintStream.

Post-processing scripts must set the **Status** property to specify the status of the script. The script can specify any value for the **Status** property. For example, to specify that the script was a success, run the following command in the post-processing script:

```
properties.put("Status","Success");
```

You can use a post-processing script to set output properties that can be used in other steps in the same process, which enables complex workflows. In a post-processing (or precondition) script, refer to prior step output properties this way:

```
properties.get("stepName/propName")
```

For example, to set a property named `myProp` whose value is included in the output log, enter this script:

```
var exit = properties.get('exitCode');

scanner.register("regex", function(lineNumber, line) {
     var thing = 'do stuff';
});
```

```
scanner.scan();

if (exit == 0) {
    properties.put('Status', 'Success');
}
else {
     properties.put('Status', 'Failure');
}
```

In instances other than post-processing and precondition scripts, refer to prior step output properties this way:

```
${p:stepName/propName}
```

The script that is defined in the `<post-processing>` element is the step's default behavior. Users can also provide their own script, overriding the default behavior, when they define the step in the process editor. See Editing processes on page 2453.

Upgrading plug-ins

This example shows how to upgrade a plug-in.

To create an upgrade, first, increment the number of the `version` attribute of the `<identifier>` element in the `plugin.xml` file. Next, update the property and step-type elements in the `plugin.xml` file.

If you change the names of a step or property, you must create an `upgrade.xml` file that contains the changes. Add a `<migrate>` element in the `upgrade.xml` file with a `to-version` attribute that contains the new number. Finally, place the names of changed property and step-type elements that match the updated `plugin.xml` file in this element, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin-upgrade
        xmlns="http://www.&company;.com/UpgradeXMLSchema_v1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <migrate to-version="3">
    <migrate-command name="Run SQLPlus script">
      <migrate-properties>
        <migrate-property name="sqlFiles" old="sqlFile"/>
      </migrate-properties>
    </migrate-command>
  </migrate>
</plugin-upgrade>
```

You can also make a script-only upgrade, which is an upgrade that contains changes to the step's associated scripts and files but does not change the `plugin.xml` file. This mechanism can be useful for plug-in development and for minor bug-fixes and updates.

If an update to a plug-in removes functionality, such as removing a step property, you must upgrade the plug-in and then open and re-save any processes that use the step. Until you open and re-save the processes, they use the pre-upgrade steps.

When creating a new version of a plugin, you many change the value of server:sub-tag in `plugin.xml`, and you do not need to add an entry to `upgrade.xml` for the change to take effect.

The `info.xml` file

Use the `info.xml` file is required for each plug-in. It describes the plug-in and provides release notes to users for each version.

The following code is an example of a simple `info.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<pluginInfo>
  <author name="Deploy User">
    <organization>IBM</organization>
    <email>ucplugin@us.ibm.com</email>
    <website>https://urbancode.github.io/IBM-UCx-PLUGIN-DOCS/UCD/</website>
    <bio/>
  </author>

  <integration type="Deploy"/>

  <source url="https://urbancode.github.io/IBM-UCx-PLUGIN-DOCS/UCD/"/>

  <licenses>
    <license type="IBM"/>
  </licenses>

  <tool-description>Sample plug-in for DevOps Deploy</tool-description>

  <related-info>
    <link description="Documentation for DevOps Deploy"
  title="Knowledge Center" type="WEBSITE"
  url="https://www.ibm.com/docs/en/urbancode-deploy"/>
  </related-info>

  <release-version>12345</release-version>

  <release-notes>
      <release-note plugin-version="2">
Fixed some bugs with the plug-in.
      </release-note>
      <release-note plugin-version="1">
Fixed some bugs with the plug-in.
      </release-note>
  </release-notes>
</pluginInfo>
```

This file contains the following sections:

`<author>` (required)

This section provides information about the author of the plug-in.

`<integration>` (required)

This tag specifies the general type of plug-in. For source configuration plug-ins, specify `Source`. For plug-ins that retrieve artifacts from a repository during a deployment, specify `Artifact`. For plug-ins that deploy code to a middleware server, specify `Deploy`. For other automation tasks, specify `Automation`.

<source>

This tag specifies the location of the source code for the plug-in, if that source code is available.

<licenses>

This tag provides a description of the license for the plug-in.

<tool-description>

This tag provides information about the software or system that the plug-in integrates with.

<related-info>

This tag provides related links.

<meta-html>

This tag provides information about the description of the plug-in on web pages.

<release-version>

The build version of the plug-in, which is not necessarily the same as the version in the `plugin.xml` file.

<release-notes> (required)

This section includes release notes for the current and previous versions of the plug-in. These versions must match the version in the `plugin.xml` file.

Automatically importing objects from plug-ins

You can add objects to plug-ins so that they are automatically imported when the plug-in is installed or upgraded. Objects include applications, components, component templates, and processes. For example, you can add a component template that implements a standard process to a plug-in, so that the plug-in user does not have to create the component template.

To automatically import objects from plug-ins, place the JSON representation of the object in the plug-in archive file, in the appropriate directory. The JSON files must have the `.json` extension. For example, if you have an application in a JSON file, the file must be in the `imports/applications` folder of the archive file.

The following table lists the locations for each object type.

Object type	Location
Applications	`imports/applications/`
Components	`imports/components/`
Component templates	`imports/componenttemplates/`
Processes	`imports/processes/`

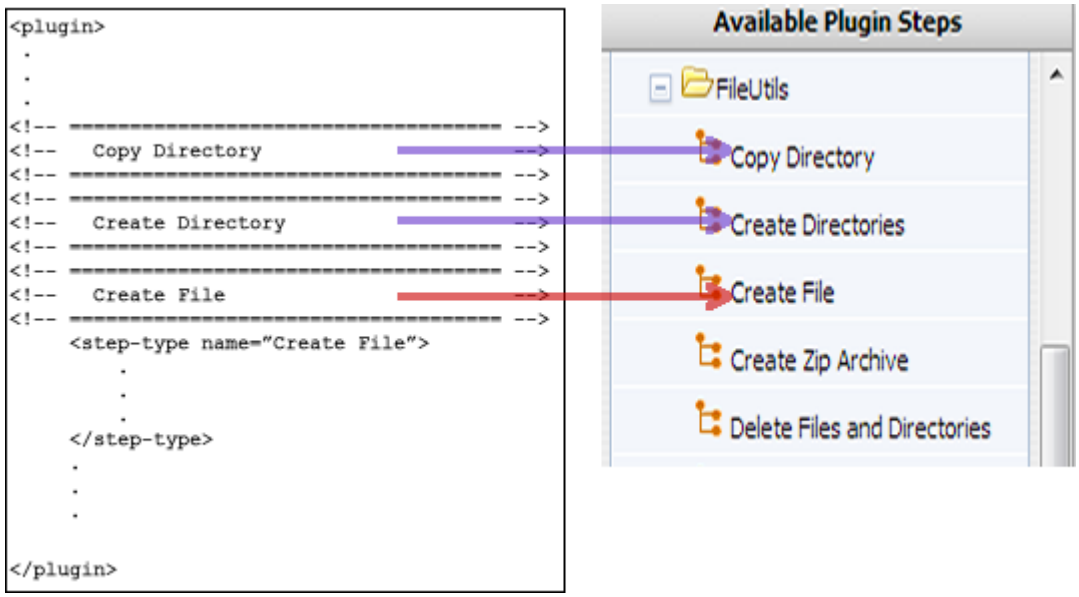
Example plug-in

A plug-in consists of one or more steps. A step represents a unit of functionality that can be user-configured and combined with other steps into a process. Creating a plug-in consists in defining its individual steps and then grouping them together for presentation in HCL DevOps Deploy (Deploy).

Plug-in steps are run by an agent that is installed in the target environment. This means that plug-ins can be written in any scripting language as long as the agent can access the required scripting tools on the host.

In this section, we examine the mechanics of plug-ins by examining a plug-in step in detail. The example plug-in that we use is FileUtils, which contains several steps that are related to file manipulation. The FileUtils plug-in is shipped with Deploy.

Each plug-in step is an individually configurable object in the editor. In the following illustration, you can see some of FileUtils' individual steps in the process editor.



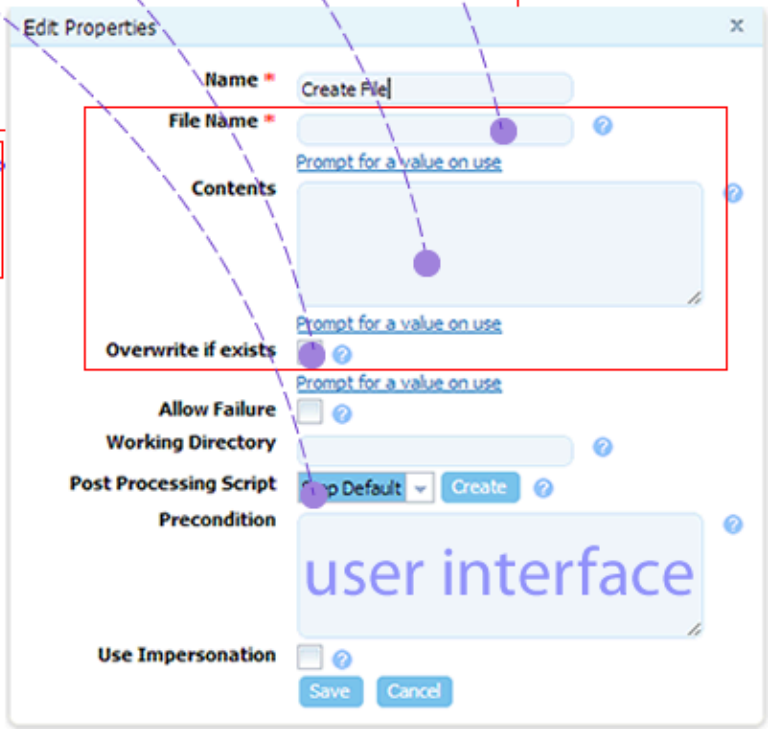
We examine the `Create File` step in this example. Create File is straightforward and (as the name implies) creates a file.

Each step (`step-type` element) has the same structure:

- `properties` element can contain any number of `<property>` child elements; property values can be supplied at design time or runtime
- `post-processing` element provides error-handling and sets property values that can be used by other steps
- `command` element performs the step's function. The function can be defined completely by the element, or be constructed in part or entirely from the step's properties at design time or runtime

The following figure illustrates the structure of the Create File step.

```
<step-type name="Create File">
     <description>Create a new file.</description>
     <properties>
       <property name="file" required="true">
         <property-ui type="textBox"
                      label="File Name"
                      description="The name of the new file."/>
       </property>
       <property name="contents">
         <property-ui type="textAreaBox"
                      label="Contents"
                      description="The contents of the new file."/>
       </property>
       <property name="overwrite">
         <property-ui type="checkBox"
                      label="Overwrite if exists"
                      description="Check here to overwrite an existing file."/>
       </property>
     </properties>
     <post-processing>
       <![CDATA[
         if (properties.get("exitCode") != 0) {
             properties.put(new java.lang.String("Status"), new java.lang.String("Failure"));
         }
         else {
             properties.put("Status", "Success");
         }
       ]]>
     </post-processing>

     <command program="${GROOVY_HOME}/bin/groovy">
       <arg file="create_file.groovy" />
       <arg file="${PLUGIN_INPUT_PROPS}"/>
       <arg file="${PLUGIN_OUTPUT_PROPS}"/>
     </command>
</step-type>
```



Step properties

In the context of our discussion, properties are values that are used by the step's command. Step properties are defined with the `property` element.

As you can see in Example plug-in on page 2580, the Create File step has three properties:

- `file` contains the name of the file the command creates; it is represented as a text box in the process designer
- `contents` contains the content of the file; it is represented by a text-area box, which can contain a large amount of data
- `overwrite` is used to specify whether the file can overwrite an existing file; it is represented by a check box in the process designer

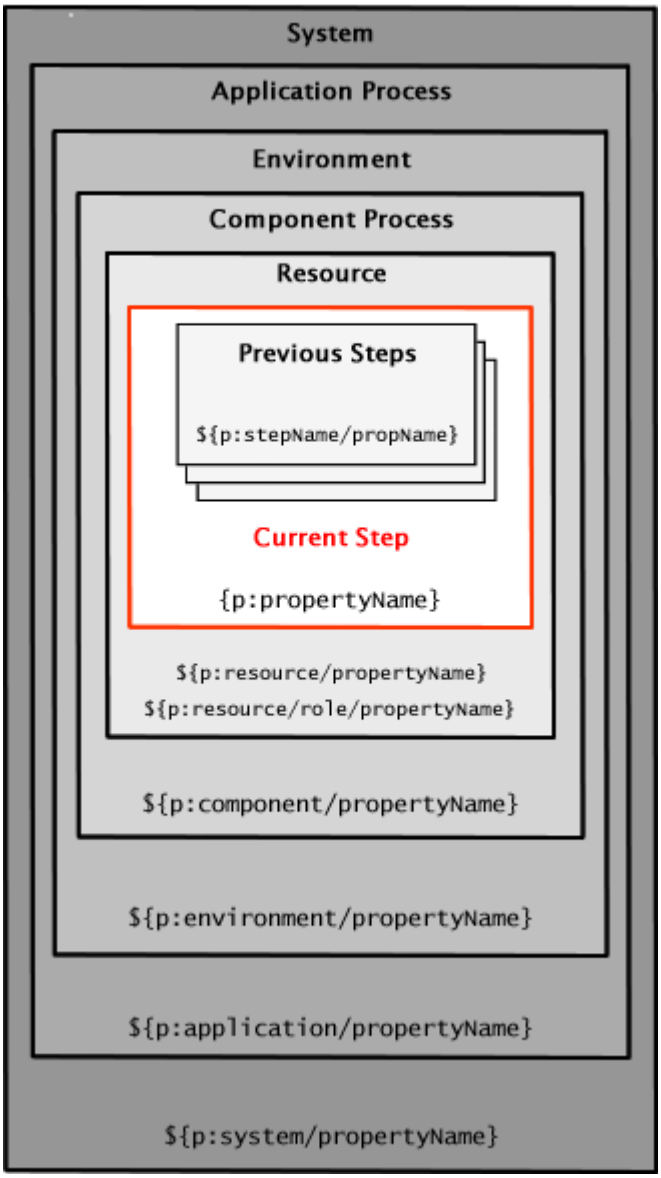
These properties are displayed in the dialog box that appears when the step is added to a component process. The other properties in the dialog are displayed for every step (post-processing is discussed below). Property values can be entered into the dialog box by the process designer at design-time, or left to be furnished at runtime by the user that runs the application.

You can configure most properties with the `property-ui` child element (the `selectBox` type requires the `value` child element as well). See Creating plug-ins on page 2561 for information about the options available for presentation in the UI. Default values can be defined when you create the step.

In addition to a step's own properties, a command has access to properties set earlier by other steps within the process, to properties set by the application that started the component process, to properties on the target environment and resource.

The following illustration shows the properties that are available to an individual step. Step property values become unavailable after the component process ends.

Properties Available to Component Steps



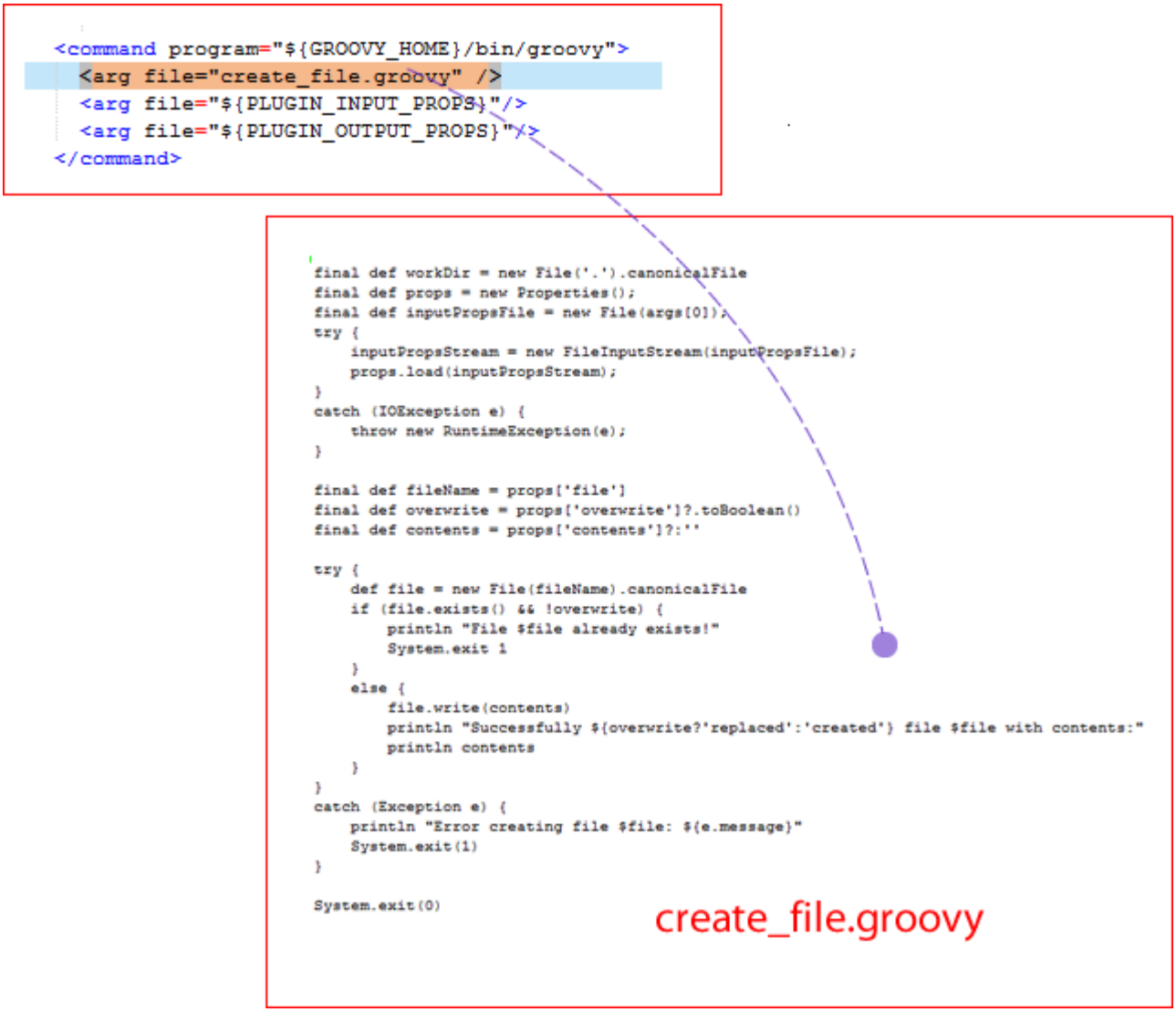
Runtime defined properties are combined with the properties that were defined earlier and together sent to the agent. Earlier properties (those properties that were defined outside the current process) are retrieved from the database. How properties are processed and consumed is discussed further in the next section.

Step commands

The step's `command` element specifies the scripting tool that runs the step and identifies the file that contains the actual script.

The agent that runs the step downloads the plug-in and expects to find the script among the downloaded files. The agent must also have access to the scripting tool. Any other arguments that are required by the script or tool can also be specified with the `arg file` attribute. The `arg file` attributes should be defined in the order compatible with the scripting tool.

The Create File command script is written in Groovy. Groovy is scripting language similar to Java™. The following illustration shows a command that runs an external Groovy file that is named `create_file.groovy`.



This command instructs the Groovy interpreter to run `create_file.groovy` (more about Groovy and the particulars of this file in the following section).

This line (which is part of every command):

```
<arg file="${PLUGIN_INPUT_PROPS}"/>
```

sends a file that contains the properties that are required by the step to the agent. The properties in the file are those properties that are furnished at runtime and others that are defined earlier that are required by the step. See Example plug-in on page 2580. The `${PLUGIN_INPUT_PROPS}` variable resolves to the location of this properties file.

And this line (which is also part of every command):

```
<arg file="${PLUGIN_OUTPUT_PROPS}"/>
```

refers to the file returned by the agent after finishing the step. The properties in this file are available to later steps in the process. The `${PLUGIN_OUTPUT_PROPS}` variable resolves to the location of this properties file.

create_file.groovy

The `create_file.groovy` file contains the Groovy script that runs the step's command.

Groovy is a dynamic scripting language (similar to Python, Ruby, and Perl) for the Java™ platform. Most Java™ code is also syntactically valid Groovy, which makes Groovy popular with Java™ programmers. Groovy provides native support for regular expressions.

The first lines of the script create a properties object, `props`. Then it attempts to load the properties from the file that is sent from the server (specified by the `${PLUGIN_OUTPUT_PROPS}` variable). If it can load the file, it populates `props`; otherwise, it throws an exception.

```
final def workDir = new File('.').canonicalFile
final def props = new Properties();
final def inputPropsFile = new File(args[0]);
try {
inputPropsStream = new FileInputStream(inputPropsFile);
props.load(inputPropsStream);
}
catch (IOException e) {
throw new RuntimeException(e);
}
```

To run the command (create a file), the script uses the properties that are defined by the step itself. The script retrieves the three properties from `props` and creates corresponding local variables.

Next, the script creates a file with a name specified by `fileName`, and tests the `overwrite` Boolean variable. If a file with the same name exists and `overwrite` is false, the script ends (fails) with an exit code of 1. Exit codes can be examined during post-processing.

Otherwise, the file is written with the content of `contents`. A message is written to the output log, and the exit code is set to 0 (success).

```
final def fileName = props['file']
final def overwrite = props['overwrite']?.toBoolean()
final def contents = props['contents']?:''

try {
def file = new File(fileName).canonicalFile
if (file.exists() && !overwrite) {
println "File $file already exists!"
System.exit 1
}
else {
file.write(contents)
println "Successfully ${overwrite?'replaced':'created'} file
$file with contents:"
println contents
}
}
```

```
catch (Exception e) {
println "Error creating file $file: ${e.message}"
System.exit(1)
}

System.exit(0)
```

Creating notification templates

Notification templates are the basis for emails that the server sends to users after an event occurs on the server. You can create customized notification templates by modifying the default templates provided, or by creating new templates of your own.

About this task

Notification templates are in Apache Velocity format. You can use this format to put together a message template with scripts and parameters that are specified when the notification happens. For example, the template can include information about a process that started, succeeded, or failed. For more information about creating templates with Velocity, see https://velocity.apache.org/.

Notifications templates are in XML format. The XML references application-specific parameters that are used by Java code to produce the notification email sent to the user. The application-specific parameters that can be used to customize the XML templates are provided in the topic Parameters for notification templates on page 2588.

1. Duplicate an existing template.
   The application templates are in the folder *application_data*`/conf/server/notification-templates`, where *application_data* is the application data folder. The default location is `/opt/install-dir/server/appdata/conf/server/notification-templates` on Linux™ and `C:\Program Files\devops-deploy\server\appdata\conf\server\notification-templates` on Windows™.
2. Open the new template in a text editor.
3. In the `<name>` tag, give the new template a unique name.
   Each template must have a unique name.
4. In the template, add parameters and other information.

   To use information from the event that created the notification, use the parameters in Parameters for notification templates on page 2588.

   Example
   For example, if you are creating a notification for an application process, you can use the parameter **$applicationProcess.getName()** to get the name of the process.
   Example

   You can also refer to properties just like you can in process steps. For example, to refer to the property **environment.name**, use the following code:

   ```
   $properties.get("environment.name")
   ```

Similarly, if the step `myStep` creates the output property `outputProperty1`, you can refer to the property in the notification template with the following code:

```
$properties.get("myStep/outputProperty1")
```

You use the same syntax to refer to other properties, too. For example, if an application has an application property that is named `myApplicationProp`, you can refer to that property with the following code:

```
$properties.get("application/myApplicationProp")
```

Similarly, you can refer to environment properties with code that is similar to the following example:

```
$properties.get("environment/myEnvironmentProp")
```

5. Restart the server.
   The server loads notification templates at start time, so when you change a template, you must restart the server before you can use it.

What to do next

Now you can use this template as part of a notification scheme. See Creating Notifications in a Notification Scheme on page 2081.

Parameters for notification templates

Depending on the context of the notification, you can insert parameters into the notification template. When the notification is sent, these parameters are replaced with values such as the names and information about applications, components, and processes.

Note: Not all parameters are available in all contexts. For example, when you run a generic process, parameters that are related to applications are not available.

Table 769. General parameters

Parameter	Description	Example
$externalUrl()	The URL of the server	`http://myserver.example.com:8443`
$request	The request that caused the notification, such as an application process request	`ApplicationProcessRequest#74ae3853-3444-4d7d-a0f3-68107f47b29f`

Table 770. Application parameters

Parameter	Description	Example
$application.getName()	The name of the application	`My application`

Table 770. Application parameters

(continued)

Parameter	Description	Example
$application.getId()	The ID of the application	`b37054b7-7b20-49b6-973d-fd0ee0b1b62b`
$application.getDescription()	The description of the application	`This is my application`
$application.getPath()	The URL path of the application	`applications/b37054b7-7b20-49b6-973d-fd0ee0b1b62b`

Table 771. Application process parameters

Parameter	Description	Example
$applicationProcess.getId()	The ID of the application process	`50d5bca2-d8d9-47f2-a0ce-5b75b3f24d3b`
$applicationProcess.getName()	The name of the application process	`Deploy application`
$appRequest.getPropSheet().getPropValue()	The name of the application process property.	`property_name`

Table 772. Environment parameters

Parameter	Description	Example
$environment.getId()	The ID of the application	`072a54df-8c3a-4b23-a660-323cf205c86c`
$environment.getName()	The name of the environment	`Environment 1`
$environment.getDescription()	The description of the environment	
$environment.getColor()	The color that is assigned to the environment in the table of environments	`#00B2EF`
$environment.isRequireApprovals()	True if the environment requires approvals	

You can retrieve an array of application process versions with the parameter **$versions**. For the size of the array, use the parameter **$versions.size()**. To access individual versions within this array, access them with a loop as in the following example:

```
<div class="data-table-container">
#if ($versions.size() > 0)
```

```
  <h3>Versions Included:</h3>
  <table class="data-table" cellpadding="4" cellspacing="1" width="100%">
   <thead class="data-table-head">
      <th style="text-align:left;" scope="col" valign="middle">Component</th>
      <th style="text-align:left;" scope="col" valign="middle">Version</th>
      <th style="text-align:left;" scope="col" valign="middle">Description</th>
   </thead>
   <tbody>
    #foreach ($version in $versions)
      #if ($velocityCount % 2 == 1)
          #set ($rowClass = "odd")
      #else
          #set ($rowClass = "even")
      #end
      <tr class="$rowClass">
       <td>$version.getComponent().getName()</td>
       <td>$version.getName()</td>
       <td>
         #if ($version.getDescription())
           $version.getDescription()
         #end
       </td>
     </tr>
    #end
   </tbody>
  </table>
#else
  <h3>No Versions Included</h3>
#end
</div>
```

Within the loop, you can access information about the individual versions with the parameters in the following table:

Table 773. Version parameters

Parameter	Description	Example
$version.getName()	The name of the component version	
$version.getComponent()	The component	

Table 774. Snapshot parameters

Parameter	Description	Example
$snapshot.getId()	The ID of the snapshot	`5d95883f-be45-4abe-8664-3ca6a7635a12`
$snapshot.getName()	The name of the snapshot	`My snapshot`
$snapshot.getDescription()	The description of the snapshot	
$snapshot.getCreatedDate()	The creation date of the snapshot	`Fri Mar 20 09:56:42 EDT 2015`

Table 774. Snapshot parameters

(continued)

Parameter	Description	Example
$snapshot.getCreatedTime()	The creation time of the snapshot in UNIX™ format	`1418136059500`

Table 775. User parameters

Parameter	Description	Example
$user.getId()	The ID of the user	`20000000000000000000000001000000`
$user.getName()	The user name	`jsmith`
$user.getActualName()	The full name of the user	`Joe Smith`
$user.getEmail()	The email address of the user	`jsmith@example.com`

Table 776. Approval parameters

Parameter	Description	Example
$processname	The name of the requested process	
$scheduledDate	The date for which the process is scheduled	`Fri Apr 10 13:16:53 EDT 2015`
$appRequest.getDescription()	The description of the application process request	
$requestType	The type of request	`applicationProcessRequest`
$roles	The roles that a user must have to approve the request	`[Role: Administrator]`
$appRequest	For application process requests, the application process request ID	`ApplicationProcessRequest#81701a8b-332c-47dd-8efe-87aab8fa2dfd`

Command-line client (CLI) reference

CLI is a command-line interface that provides access to the HCL DevOps Deploy (Deploy) server. It can be used to find or set properties and to run numerous functions.

The command-line client contains commands that match the REST commands for the Deploy server.

Installing the command-line client

The command-line client is available in an archive file that you can download from the server.

Before you begin

Make sure that the **JAVA_HOME** system variable is set to the location of a JRE or JVM.

About this task

Beginning with Deploy version 6.1, the command-line client is installed on agents automatically. The client is in the folder *agent_install*`/opt/udclient`.

1. To install the tool, download the `udclient.zip` file to the system on which you want to use the tool.
   You can download the file from the Deploy server by clicking **Help > Tools** and then clicking Deploy **Client**. You can also find the file in the folder *server_install*`/opt/tomcat/webapps/ROOT/tools/`, where *server_install* is the installation folder for the server.
2. Extract the file.
3. **Optional:** Set system variables such as **DS_WEB_URL** so you do not have to specify credentials each time you run a command.
   See Command format on page 2592.

What to do next

For information about specific commands, see CLI Commands on page 2595.

Command format

To run a command, open a command-line window and specify the command and its parameters. In some cases, you must pass a JSON string with information for a command.

Commands have the following format:

```
udclient [global-args...] [global-flags...] <command> [args...]
```

Notes:

- You must enter CLI commands on one line.
- CLI commands and parameters are case-sensitive.
- Using CLI commands requires the same permissions as using the web interface. For information about permissions, see Roles and permissions on page 2187.

Authenticating

You can authenticate to the server in any of the following ways:

- You can specify a user name and password in the **username** and **password** parameters, and the location of the server in the **weburl** parameter, as shown in the following examples.
- You can specify a user name and password in the **DS_USERNAME** and **DS_PASSWORD** system variables, and the location of the server in the **DS_WEB_URL** system variable.
- You can specify a token in the **authtoken** parameter or in the **DS_AUTH_TOKEN** system variable.
- You can use the login on page 2842 command to create a temporary authentication to the server.

Passing parameters

Most commands require one or more parameters. Parameters can change how the server interprets the command.

For example, if you do not pass any parameters to the getResources command (except the required global parameters), the server lists the top-level resources:

```
udclient -weburl http://ucdserver.example.com:8080
  -username admin
  -password admin
  getResources
```

However, if you pass the path or ID of a resource in the **parent** parameter, the command returns the resources that are under that resource, as in the following example:

```
udclient -weburl http://ucdserver.example.com:8080
  -username admin
  -password admin
  getResources
  -parent "/My Agents"
```

Passing JSON strings

Some commands require a JSON string or file instead of or in addition to the parameters.

For example, the createApplication command creates an application. To use this command, you must pass a valid JSON string that specifies the name, description, and some properties of the new application. The JSON string for this command must follow this template:

```
{
  "description": "Description",
  "enforceCompleteSnapshots": "Specify true to require
     an explicit version for each component",
  "name": "Application name or ID",
  "notificationScheme": "Notification scheme"
}
```

Note: To see the JSON template that a command requires, run the command with the `-t` switch, as in the following example:

```
udclient -weburl http://ucdserver.example.com:8080
  -username admin
  -password admin
```

```
createApplication
-t
```

This template is also listed in the reference information for the command; see createApplication on page 2636.

For example, the following JSON string represents an application that is named `My Application`:

```
{
  "description": "My new application",
  "enforceCompleteSnapshots": "false",
  "name": "My Application",
  "notificationScheme": "Default Notification Scheme"
}
```

To pass this JSON string, save the string to a file and pass the file to the command. For example, if you save the string to a file that is named `newApplication.json`, the command looks like the following example:

```
udclient -weburl http://ucdserver.example.com:8080
  -username admin
  -password admin
  createApplication
  newApplication.json
```

Note: The string that you pass to the command must be a valid JSON string.

Global parameters and flags

The following table lists the global parameters for the commands. These global parameters are listed as `[global-args...]` in code examples.

Table 777. Global parameters

Argument	Description
-authtoken, **--authtoken**	Optional. Can be set with the **DS_AUTH_TOKEN** environment variable. An authentication token (authtoken) that is generated by the server. Either an authtoken or a user name and password are required.
-password, **--password**	Optional. Can be set with the **DS_PASSWORD** environment variable. A password to authenticate with the server. Either an authtoken or a user name and password are required.
-proxyHost, **--proxyHost**	Optional. Can be set with the **proxyHost** environment variable. The host name of a proxy (for example, an agent relay) to use instead of a direct connection to the server.
-proxyPassword, **--proxyPassword**	Optional. Can be set with the **proxyPassword** environment variable. A password to use to authenticate with the proxy.
-proxyPort, **--proxyPort**	Optional. Can be set with the **proxyPort** environment variable. The proxy port to use if a proxy host name is specified.

Table 777. Global parameters (continued)

Argument	Description
-proxyUser, **--proxyUser**	Optional. Can be set with the **proxyUser** environment variable. A user name to use to authenticate with the proxy.
-username, **--username**	Optional. Can be set with the **DS_USERNAME** environment variable. A user name to authenticate with the server. Either an authtoken or a user name and password are required.
-weburl, **--weburl**	Required. Can be set with the **DS_WEB_URL** environment variable. The base URL of the Deploy server. For example, http://ds.example.com:8585.

The following table shows the global flags. These global flags are listed as `[global-flags...]` in code examples.

Table 778. Global flags

Flag	Description
-t, **--getTemplate**	Shows the JSON template for the command instead of running the command. If a file argument is provided, the template is output to the specified file.
-h, **--help**	Prints the full description and help of the command instead of running the command. You can get the information, such as: • The name of the command and its usage. • The synopsis of the command. • The required and optional parameters you can use with the command. • For some commands, the possible HTTP response status codes.
-v, **--verbose**	Prints extra information.

When you specify objects such as applications, components, agents, and resources on the command line, you can use the name of the object or the object ID. The ID of an object is part of the URL that is displayed when you inspect the object in Deploy. For example, if you browse to a specific component and the URL is `https://ucdeploy.example.com:8443/#component/2f153e93-00c3-44ef-a7fd-87e544e698d4`, then the ID of the component is `2f153e93-00c3-44ef-a7fd-87e544e698d4`.

This example shows the getResources command:

```
udclient -weburl http://localhost:8080 -username admin -password admin getResources
```

CLI Commands

The following list shows the commands that are available.

Note:

- CLI commands do not support new lines. Entries are broken for display purposes only.
- While you create standard-type objects such as components, applications, environments, or resources, you must not specify a type. For example, you can omit the resourceRoleLabel attribute.

Most of these commands have equivalent commands in the REST API. See REST commands for the DevOps Deploy server.

addAgentPoolToTeam

Add an agent pool to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addAgentPoolToTeam [parameters]
```

Parameters

Table 779. Parameters

Parameter	Type	Required	Description
agentPool	string	true	Name or ID of the agent pool
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard agent pool".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addAgentPoolToTeam
  -agentPool MyAgentPool
  -team MyTeam
```

Related REST command: Add an agent pool to a team.

addAgentServerPin

Add public key pin for a server host to an agent

This command requests the agent add a public key pin for a server host, but does not wait for or report the result. The returned maintenance task can be used to query the result.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addAgentServerPin [parameters]
```

Parameters

Table 780. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
host	string	true	The host name, as it is known by the agent, that will receive the public key pin
alias	string	true	A keypair alias in the server keystore file that holds the target public key
allowUnknownHost	boolean	false	If true, allow setting a pin for a host not currently known to the agent (optional)

Related REST command: Add public key pin for a server host to an agent.

addAgentToPool

Add an agent to an agent pool

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addAgentToPool [parameters]
```

Parameters

Table 781. Parameters

Parameter	Type	Required	Description
pool	string	true	Name or ID of the agent pool
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addAgentToPool
  -agent Agent1
  -pool Pool1
```

Related REST command: Add an agent to an agent pool.

addAgentToTeam

Add an agent to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addAgentToTeam [parameters]
```

Parameters

Table 782. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Application".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addAgentToTeam
  -agent MyAgent
  -team MyTeam
```

Related REST command: Add an agent to a team.

addApplicationToTeam

Add an application to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addApplicationToTeam [parameters]
```

Parameters

Table 783. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
team	string	true	Name or ID of the team

Table 783. Parameters (continued)

Parameter	Type	Required	Description
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Application".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addApplicationToTeam
  -application JPetStore
  -team MyTeam
```

Related REST command: Add an application to a team.

addComponentResourcePropDef

Create a component resource property definition

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addComponentResourcePropDef [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "allowedValues": "Comma separated list of allowed values
  in SELECT or MULTI_SELECT properties (Optional)",
  "component": "Name or ID of the component",
  "description": "Description for the property (Optional)",
  "label": "Label of the property (Optional)",
  "name": "Name of the property to set",
  "pattern": "A regular expression that specifies valid
  values for the property. Follows java.util.regex.Pattern
  (Optional)",
  "required": "Whether the property is required (Optional,
  Boolean)",
  "type": "the type of property definition; valid values
  are TEXT (the default), , TEXTAREA, CHECKBOX, SELECT,
  MULTI_SELECT, DATETIME, and SECURE)",
  "value": "Default value of the property definition
  (Optional)"
}
```

Related REST command: Create a component resource property definition.

addComponentTemplateToTeam

Add a component template to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addComponentTemplateToTeam [parameters]
```

Parameters

Table 784. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Component Template".

Related REST command: Add a component template to a team.

addComponentToApplication

Add a component to an application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addComponentToApplication [parameters]
```

Parameters

Table 785. Parameters

Parameter	Type	Required	Description
component	string	true	Name of the component
application	string	true	Name of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addComponentToApplication
  -application JPetStore
  -component JPetStore-WEB
```

Related REST command: Add a component to an application..

addComponentToTeam

Add a component to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addComponentToTeam [parameters]
```

Parameters

Table 786. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Component".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addComponentToTeam
  -component JPetStore-APP
  -team MyTeam
```

Related REST command: Add a component to a team.

addEnvironmentBaseResource

Add a base resource to an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addEnvironmentBaseResource [parameters]
```

Parameters

Table 787. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name of the application; this value is required if you specify the environment name instead of ID
resource	string	true	Path or ID of the resource to add

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addEnvironmentBaseResource
  -application MyApplication
  -environment Environment1
  -resource /agents/agent1
```

Related REST command: Add a base resource to an environment.

addEnvironmentProperty

Create a component environment property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addEnvironmentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Name or ID of the component",
  "default": "Default value (Optional)",
  "description": "Description for the property (Optional)",
  "label": "Label of the property (Optional)",
  "name": "Name of the property",
  "pattern": "A regular expression that specifies valid
  values for the property. Follows java.util.regex.Pattern
  (Optional)",
```

```
  "required": "Whether the property is required (Optional,
  Boolean)",
  "secure": "Specify true to create a secure property
  (Optional, Boolean)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addEnvironmentProperty
  -name MyProperty
  -component d2e59ca4-a2f5-4851-9c85-629c1019ba20
  -required true
```

Related REST command: Create a component environment property.

addEnvironmentToTeam

Add an environment to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addEnvironmentToTeam [parameters]
```

Parameters

Table 788. Parameters

Parameter	Type	Required	Description
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
environment	string	true	Name or ID of the environment
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Environment".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addEnvironmentToTeam
```

```
  -application MyApplication
  -environment MyEnvironment
  -team MyTeam
```

Related REST command: Add an environment to a team.

addGroupToTeam

Add a group to a role on a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addGroupToTeam [parameters]
```

Parameters

Table 789. Parameters

Parameter	Type	Required	Description
group	string	true	Name or ID of the group
team	string	true	Name or ID of the team
role	string	true	Name or ID of the role to give to the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addGroupToTeam
  -group group1
  -team MyTeam
  -role MyRole
```

Related REST command: Add a group to a role on a team.

addJavaPackage

Upload a Java package.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addJavaPackage [parameters]
```

Parameters

Table 790. Parameters

Parameter	Type	Required	Description
name	string	false	Custom package name (typically, the file name of the package). Limited to 100 characters from [-_.0-9a-zA-Z]. Certain names that have special meaning as file names such as ".." or "COM1" are rejected.

Related REST command: Upload a Java package..

addPostProcessingScriptToTeam

Add a post processing script to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addPostProcessingScriptToTeam [parameters]
```

Parameters

Table 791. Parameters

Parameter	Type	Required	Description
script	string	true	Name/ID of the post processing script
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Post Processing Script"

Related REST command: Add a post processing script to a team.

addProcessToTeam

Add a process to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addProcessToTeam [parameters]
```

Parameters

Table 792. Parameters

Parameter	Type	Required	Description
process	string	true	Name or ID of the process
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Process".

Related REST command: Add a process to a team.

addResourceToTeam

Add a resource to a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addResourceToTeam [parameters]
```

Parameters

Table 793. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Resource".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addResourceToTeam
  -resource /Resource1/Subresource1
  -team MyTeam
```

Related REST command: Add a resource to a team.

addStatusToSnapshot

Add a status to a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addStatusToSnapshot [parameters]
```

Parameters

Table 794. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot.
application	string	false	Name or ID of the application; this value is required if you specify the snapshot name instead of the ID.
statusName	string	true	Name or ID of the status. This value cannot be longer than 36 characters.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addStatusToSnapshot
  -application MyApplication
  -snapshot MySnapshot
  -statusName NewStatus
```

Example

Example response

```
{
  "id": "d3dfd0c8-90f5-427b-8432-3c71c086cc93",
  "name": "MySnapshot",
  "description": "",
  "created": 1450130946382,
  "active": true,
  "versionsLocked": false,
  "configLocked": false,
  "applicationId": "99822240-cc96-4e4e-91e9-bcc483521cec",
  "user": "jsmith"
}
```

Related REST command: Add a status to a snapshot.

addTagToAgent

Add a tag to an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addTagToAgent [parameters]
```

Parameters

Table 795. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
tag	string	true	Name of the tag
color	string	false	Hex representation of the color for the tag, such as FF0000. This parameter is used only if the tag is being created.
description	string	false	Description of the tag. This parameter is used only if the tag is being created.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addTagToAgent
  -agent myAgent
  -tag Tag1
```

Related REST command: Add a tag to an agent.

addTagToApplication

Add a tag to an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addTagToApplication [parameters]
```

Parameters

Table 796. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
tag	string	true	Name of the tag
color	string	false	Hex representation of the color for the tag, such as FF0000. This parameter is used only if the tag is being created.
description	string	false	Description of the tag. This parameter is used only if the tag is being created.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addTagToApplication
  -application JPetStore
  -tag Tag1
```

Related REST command: Add a tag to an application.

addTagToComponent

Add a tag to a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addTagToComponent [parameters]
```

Parameters

Table 797. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
tag	string	true	Name of the tag

Table 797. Parameters (continued)

Parameter	Type	Required	Description
color	string	false	Hex representation of the color for the tag, such as FF0000. This parameter is used only if the tag is being created.
description	string	false	Description of the tag. This parameter is used only if the tag is being created.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addTagToComponent
  -component JPetStore-APP
  -tag Tag1
```

Related REST command: Add a tag to a component.

addTagToResource

Add a tag to a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addTagToResource [parameters]
```

Parameters

Table 798. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
tag	string	true	Name of the tag

Table 798. Parameters (continued)

Parameter	Type	Required	Description
color	string	false	Hex representation of the color for the tag, such as FF0000. This parameter is used only if the tag is being created.
description	string	false	Description of the tag. This parameter is used only if the tag is being created.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addTagToResource
  -resource /Resource1/Subresource1
  -tag Tag1
```

Related REST command: Add a tag to a resource.

addUserToGroup

Add a user to a group

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addUserToGroup [parameters]
```

Parameters

Table 799. Parameters

Parameter	Type	Required	Description
user	string	true	Name of the user
group	string	true	Name of the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addUserToGroup
  -user jsmith
  -group group1
```

Related REST command: Add a user to a group.

addUserToTeam

Add a user to a role on a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addUserToTeam [parameters]
```

Parameters

Table 800. Parameters

Parameter	Type	Required	Description
user	string	true	Name or ID of the user
team	string	true	Name or ID of the team
role	string	true	Name or ID of the role to give to the user

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addUserToTeam
  -user jdoe
  -team MyTeam
  -role MyRole
```

Related REST command: Add a user to a role on a team.

addVersionFiles

Upload files to a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addVersionFiles [parameters]
```

Parameters

Table 801. Parameters

Parameter	Type	Required	Description
component	string	false	Name/ID of the component (Only required if not using version ID)
version	string	true	Name/ID of the version
base	string	true	Local base directory for upload. All files inside this will be sent.
offset	string	false	Target path offset (the directory in the version files to which these files should be added)
include	string	false	An include file pattern for selecting files to add (may be repeated)
exclude	string	false	An exclude file pattern for excluding files (may be repeated). Overrides includes.
saveExecuteBits	boolean	false	Saves execute bits for files.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addVersionFiles
  -component JPetStore-WEB
  -version 1.1
  -base newImageFolder
  -exclude */subDirName/*
```

addVersionLink

Add a link on a version

This command adds an external link on the specified component version.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addVersionLink [parameters]
```

Parameters

Table 802. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component. This value is required if you specify a version name instead of an ID
version	string	true	Name or ID of the version
linkName	string	true	Name of the link to add
link	string	true	Value of the link to add
isPriority	boolean	false	Optional. If true, prioritize the link by placing it in the version page header for quicker access

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addVersionLink
  -component JPetStore-APP
  -version 1.0
  -linkName "Link to IBM web site"
  -link http://www.ibm.com
```

Related REST command: Add a link on a version.

addVersionLinkWithName

Add a link on a version

This command adds an external link on the specified component version.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addVersionLinkWithName [parameters] [JSON file]
```

Parameters

Table 803. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component. This value is required if you specify a version name instead of an ID

Table 803. Parameters (continued)

Parameter	Type	Required	Description
version	string	true	Name or ID of the version

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "isPriority": "Optional. If true, prioritize the link by
  placing it in the version page header for quicker access",
  "link": "Link url",
  "name": "Link name"
}
```

Related REST command: Add a link on a version.

addVersionStatus

Add a status to a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addVersionStatus [parameters]
```

Parameters

Table 804. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version
status	string	true	Name of the status to apply

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addVersionStatus
  -component JPetStore-APP
```

```
  -version 1.0
  -status Approved
```

Related REST command: Add a status to a version.

addVersionToSnapshot

Add a component version to a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addVersionToSnapshot [parameters]
```

Parameters

Table 805. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID
version	string	true	Name or ID of the component version
component	string	false	Name of the component; this value is required if you specify the component name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  addVersionToSnapshot
  -application MyApplication
  -component MyComponent
  -snapshot MySnapshot
  -version 1.2
```

Related REST command: Add a component version to a snapshot.

addWebhookToApplication

Add a webhook to an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addWebhookToApplication [parameters]
```

Parameters

Table 806. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	ID or name of webhook template
url	string	true	Url for the webhook
environment	string	true	ID of environment to add the webhook to. This can be repeated for multiple environments.
webhookProxyHost	string	false	Host name of http proxy. Default ${p?:system/proxyHost}.
webhookProxyPort	string	false	Port of http proxy. Default ${p?:system/proxyPort}.
webhookProxyUsername	string	false	Username for authentication with http proxy. Default ${p?:system/proxyUsername}.
webhookProxyPassword	string	false	Password for authentication with http proxy. Default ${p?:system/proxyPassword}.
application	string	true	ID or name of application

Related REST command: Add a webhook to an application.

addWebhookToApplicationWithCredentials

Add a webhook to an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addWebhookToApplicationWithCredentials [JSON file]
```

Parameters

Table 807. Parameters

Parameter	Type	Required	Description
application	string	true	ID or name of application

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "environments": ["Name or ID of environments"],
  "proxyHost": "Proxy Host. Updating proxy host will unset
  proxy username and password if those are not provided",
  "proxyPassword": "Password to use for basic
  authentication with the proxy",
  "proxyPort": "Proxy Port. Updating proxy port will unset
  proxy username and password if those are not provided",
  "proxyUsername": "Username to use for basic
  authentication with the proxy",
  "url": "Url for the webhook",
  "webhookTemplate": "Name or ID of the webhook template"
}
```

Related REST command: Add a webhook to an application.

addWebhookToProcess

Add a webhook to a process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addWebhookToProcess [parameters]
```

Parameters

Table 808. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	Id or name of webhook template.
url	string	true	Url for the webhook
webhookProxyHost	string	false	Host name of http proxy. Default ${p?:system/proxyHost}.

Table 808. Parameters (continued)

Parameter	Type	Required	Description
webhookProxyPort	string	false	Port of http proxy. Default ${p?:system/proxyPort}.
webhookProxyUsername	string	false	Username for authentication with http proxy. Default ${p?:system/proxyUsername}.
webhookProxyPassword	string	false	Password for authentication with http proxy. Default ${p?:system/proxyPassword}.
process	string	true	Id or name of process

Related REST command: Add a webhook to a process.

addWebhookToProcessWithCredentials

Add a webhook to a process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addWebhookToProcessWithCredentials [JSON file]
```

Parameters

Table 809. Parameters

Parameter	Type	Required	Description
process	string	true	Id or name of process

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "proxyHost": "Proxy Host. Updating proxy host will unset
  proxy username and password if those are not provided",
  "proxyPassword": "Password to use for basic
  authentication with the proxy",
  "proxyPort": "Proxy Port. Updating proxy port will unset
  proxy username and password if those are not provided",
  "proxyUsername": "Username to use for basic
  authentication with the proxy",
  "url": "Url for the webhook",
  "webhookTemplate": "Name or ID of the webhook template"
```

```
}
```

Related REST command: Add a webhook to a process.

addZInventoryEntries

Add entries to Z inventory.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  addZInventoryEntries [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or Id of the application.",
  "component": "Name or Id of the component.",
  "deploy-time": "Time in milliseconds.",
  "environment": "Name or Id of the environment.",
  "resources": [{
    "deploy-type": "Type of deployment for this resource.",
    "inputs": "Inputs",
    "name": "Filename of the entry.",
    "operation-type": "Type of operation for this entry. e.
  g. 'Deploy' or 'Delete'.",
    "target": "Target",
    "target-type": "Type of the target for this resource."
  }],
  "version": "Name or Id of the version."
}
```

Related REST command: Add entries to Z inventory..

archiveVersion

Archive the given versions of a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  archiveVersion [parameters]
```

Parameters

Table 810. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
version	string	true	Name or ID of the version(s) to be archived. This option can be repeated to archive multiple versions.
path	string	false	Deprecated. Archive path. An absolute path which overrides the System's archive path. Must be enabled on the server.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  archiveVersion
  -component MyComponent
  -version 1.0
  -version 1.2
  -version 1.3
```

Related REST command: Archive the given versions of a component.

autoDiscoverResource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  autoDiscoverResource [parameters]
```

Parameters

Table 811. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1

Related REST command: Autodiscover resource.

bulkAddAgentServerPin

Add public key pins for server hosts to an agents in bulk

This command requests a set of agent add a public key pin for a server host, but does not wait for or report the results. The returned maintenance tasks can be used to query the result. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkAddAgentServerPin [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "alias": "A keypair alias in the server keystore file
  that holds the target public key",
    "allowUnknownHost": "If true, allow setting a pin for
  a host not currently known to the agent (optional)",
    "host": "The host name, as it is known by the agent,
  that will receive the public key pin"
  }]
}
```

Related REST command: Add public key pins for server hosts to an agents in bulk.

bulkGetAgentJavaHomes

Get list of managed Java installations in bulk

This command returns a list of managed Java installations for a set of agents. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkGetAgentJavaHomes [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
```

```
  "entity": [{"agent": "Name or ID of an agent"}]
}
```

Related REST command: Get list of managed Java installations in bulk.

bulkGetPreferredJavaPackageForAgent

Get the preferred Java packages compatible with agents in bulk.

This command gets the preferred Java packages compatible with the platforms of a set of agents. If no such package is currently defined for an agent, then no content is returned for the respective sub-request. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkGetPreferredJavaPackageForAgent [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{"agent": "Name or ID of the agent"}]
}
```

Related REST command: Get the preferred Java packages compatible with agents in bulk..

bulkInstallAgentJava

Install Java packages as a new managed Java installations in bulk

This command request a set of agents install Java, but does not wait for or report the results. The returned maintenance tasks can be used to query the result. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkInstallAgentJava [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "package": "Name or ID of the Java package"
  }]
}
```

Related REST command: Install Java packages as a new managed Java installations in bulk.

bulkRemoveAgentServerPin

Remove public key pins for server hosts from agents in bulk

This command requests a set of agent remove a public key pin for a server host, but does not wait for or report the results. The returned maintenance tasks can be used to query the result. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkRemoveAgentServerPin [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "host": "The host name, as it is known by the agent,
  from which the public key pin is removed",
    "publicKeyFingerprint": "The ingerprint for the public
  key pin to remove"
  }]
}
```

Related REST command: Remove public key pins for server hosts from agents in bulk.

bulkSetAgentJavaHome

Set the configured JAVA_HOME of agents in bulk

This command requests a set of agents set JAVA_HOME, but does not wait for or report the results. Each new JAVA_HOME will not take effect until the respective agent is restarted. The returned maintenance tasks can be used to query the results.Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkSetAgentJavaHome [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "javahome": "Name of managed Java install or absolute
  path of a valid JAVA_HOME directory"
  }]
}
```

Related REST command: Set the configured JAVA_HOME of agents in bulk.

bulkSetAgentKeypairPolicy

Add public key pins for server hosts to an agents in bulk

This command requests a set of agents set or modify their keypair policy, but does not wait for or report the results. The returned maintenance tasks can be used to query the result. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests. If the maximum age of a keypair is changed to be less than its current age, keypair rotation may begin immediately, but the outcome of the returned maintenance tasks do not depend on that.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkSetAgentKeypairPolicy [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "maxAgeDays": "The maximum age in days of the agent's
  keypair, or zero to disable rotation"
  }]
}
```

Related REST command: Add public key pins for server hosts to an agents in bulk.

bulkUninstallAgentJava

Uninstall a managed Java installations in bulk

This command requests a set of agents uninstall Java, but does not wait for or report the result. If an installation is the active JAVA_HOME or configured as JAVA_HOME, that installation is not removed. The returned maintenance tasks can be used to query the result. Individual sub-requests may succeed or fail independently. A single request may contain up to 250 sub-requests.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  bulkUninstallAgentJava [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "agent": "Name or ID of an agent",
    "package": "Name of a managed Java package"
  }]
}
```

Related REST command: Uninstall a managed Java installations in bulk.

cancelApplicationPipelineExecution

Cancel an application pipeline execution

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelApplicationPipelineExecution [parameters]
```

Parameters

Table 812. Parameters

Parameter	Type	Required	Description
executionId	string	true	Id of the pipeline execution.

Related REST command: Cancel an application pipeline execution.

cancelApplicationProcessRequest

Cancel an application process request

Cancel an outstanding application process request

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelApplicationProcessRequest [parameters]
```

Parameters

Table 813. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the application process request.

Related REST command: Cancel an application process request.

cancelComponentProcessRequest

Cancel an component process request

Cancel an outstanding component process request

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelComponentProcessRequest [parameters]
```

Parameters

Table 814. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the component process request.

Related REST command: Cancel an component process request.

cancelMaintenanceBlackoutSeries

Cancels the entire series for a recurring maintenance blackout.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelMaintenanceBlackoutSeries [parameters]
```

Parameters

Table 815. Parameters

Parameter	Type	Required	Description
series	string	true	ID of the series

Related REST command: Cancel the entire series for a recurring maintenance blackout.

cancelMaintenanceBlackoutSeriesEvent

Cancels a single event in the series for a recurring maintenance blackout.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelMaintenanceBlackoutSeriesEvent [parameters]
```

Parameters

Table 816. Parameters

Parameter	Type	Required	Description
series	string	true	ID of the series
event	string	true	ID of the event
scheduled	string	true	Scheduled date of the event (its start date)

Related REST command: Cancel a single event in the series for a recurring maintenance blackout.

cancelProcessRequest

Cancel an process request

Cancel an outstanding process request

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelProcessRequest [parameters]
```

Parameters

Table 817. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the process request.

Related REST command: Cancel a process request.

cancelScheduledDeploymentSeries

Cancels the entire series for a recurring scheduled deployment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelScheduledDeploymentSeries [parameters]
```

Parameters

Table 818. Parameters

Parameter	Type	Required	Description
series	string	true	ID of the series

Related REST command: Cancel the entire series for a recurring scheduled deployment.

cancelScheduledDeploymentSeriesEvent

Cancels a single event in the series for a recurring scheduled deployment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelScheduledDeploymentSeriesEvent [parameters]
```

Parameters

Table 819. Parameters

Parameter	Type	Required	Description
series	string	true	ID of the series
event	string	true	ID of the event

Table 819. Parameters (continued)

Parameter	Type	Required	Description
scheduled	string	true	Scheduled date of the event

Related REST command: Cancel a single event in the series for a recurring scheduled deployment.

cancelServerKeystoreImportTask

Cancels an incomplete keystore import task. Incomplete tasks can be cancelled by any user with the required permission.

Cancel an incomplete keystore import task

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelServerKeystoreImportTask [parameters]
```

Parameters

Table 820. Parameters

Parameter	Type	Required	Description
task	string	true	ID of an import task.

Related REST command: Cancel an incomplete keystore import task..

cancelWorkflow

Cancel a workflow in progress

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  cancelWorkflow [parameters]
```

Parameters

Table 821. Parameters

Parameter	Type	Required	Description
workflow	string	true	ID of the workflow to cancel.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  cancelWorkflow
  -workflow 193e892f-677d-4534-9763-ff8d005a5829
```

Related REST command: Cancel a workflow in progress.

closeTask

Approve or reject an approval task

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  closeTask [JSON file]
```

Parameters

Table 822. Parameters

Parameter	Type	Required	Description
taskId	string	true	The ID of the task.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "comment": "Comment to close the task with",
  "p:Property name": "Property value (Optional,
  Repeatable)",
  "passFail": "Submit passed to approve, fail to fail"
}
```

Related REST command: Approve or reject an approval task.

completeServerKeystoreImportTask

Complete a server keystore import task

Complete a server keystore import task by uploading a keystore containing new key pairs. This will modify keystore file at path appdata/conf/server.keystore. The uploaded keystore must be in a format consumable by your Java virtual machine; currently, JKS and PKCS12 are the supported formats. Your Java virtual machine may impose additional restrictions such that keystores from other virtual machines or generated by other tools may not be readable. Imports can only be completed by the same user that created the import task.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  completeServerKeystoreImportTask [parameters]
```

Parameters

Table 823. Parameters

Parameter	Type	Required	Description
task	string	true	ID of a prepared import task to complete

Related REST command: Complete a server keystore import task.

copyResource

Copy a resource into a new parent

To update a resource's parent with this command, you must specify, at a minimum, the resource path or id, and optionally provide info for the resource template, blueprint, and application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  copyResource [parameters] [JSON file]
```

Parameters

Table 824. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
parent	string	true	The path or id of the new parent
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agent": "Name or ID of an agent to use for automation
  (Optional)",
  "agentPool": "Name or ID of an agent pool to use for
```

```
  automation (Optional)",
  "component": "Name or ID of a component to associate
  with the resource (Optional)",
  "componentTag": "Name or ID of a component tag to
  associate with the resource; any components with that tag
  can be deployed to this resource (Optional)",
  "description": "Description",
  "impersonationForce": "Specify true to prevent children
  or steps from specifying their own impersonation settings
  (Optional)",
  "impersonationGroup": "Group for default impersonation
  (Optional)",
  "impersonationPassword": "Password for default
  impersonation. Agents running on Linux and Unix machines
  will ignore this value (Optional)",
  "impersonationUseSudo": "Specify true to use the sudo
  command for impersonation. Agents running on Windows
  machines will ignore this value (Optional)",
  "impersonationUser": "User name for default
  impersonation (Optional)",
  "inheritTeam": "Whether or not to set inherit teams for
  resource. teamMappings will be ingored if true. Defaults
  to true (Optional)",
  "name": "Name for the new resource; leave blank to
  inherit the name from an agent, pool, or role",
  "parent": "ID or path to parent resource; leave blank to
  create a root resource (Optional)",
  "role": "Name or ID of a resource role to associate with
  the agent; if this resource is for a component, specify
  the component name instead (Optional)",
  "roleProperties": {"Property name": "Property value
  (Optional)"},
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }]
}
```

Related REST command: Copy a resource into a new parent.

copyVersion

Copy a component version with all of its properties

Do not use the keywords latest, latestVersion, or newest as the version name. Artifacts are not copied to the created component version.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  copyVersion [parameters]
```

Parameters

Table 825. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the new version
from	string	true	Name of the version to copy
description	string	false	Description of the new version
type	string	false	Type to use for the new version (incremental or full); the default is the default version type for the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  copyVersion
  -component MyComponent
  -name NewCopiedVersion
  -from ExistingVersion
  -type full
```

Example

Example response


```
{
  "properties": [],
  "propSheets": [{
    "properties": [],
    "id": "62ee0d83-97aa-4006-b4a7-933a803abdc3",
    "path": "components\/5ad2f8a2-88d3-4f55-94e0-499bdbc5f0a9\/versions\/2d9e369b-8ac9-45c0-a192-017c201820e6\/
    propSheetGroup\/propSheets\/62ee0d83-97aa-4006-b4a7-933a803abdc3",
    "version": 1,
    "versionCount": 1,
    "commit": 0,
    "versioned": true,
    "propSheetDef": {
```

```
      "id": "bce76bb6-b660-4f62-b23a-859cd3ac61c2",
      "path": "components\/5ad2f8a2-88d3-4f55-94e0-499bdbc5f0a9\/versionPropSheetDef",
      "version": 1,
      "versionCount": 1,
      "commit": 9,
      "resolveHttpValuesUrl": "property\/propSheetDef\/
      components%265ad2f8a2-88d3-4f55-94e0-499bdbc5f0a9%26versionPropSheetDef.-1\/resolveHttpValues",
      "versioned": true
    }
  }],
  "totalSize": 0,
  "totalCount": 0,
  "id": "2d9e369b-8ac9-45c0-a192-017c201820e6",
  "name": "NewCopiedVersion",
  "type": "FULL",
  "created": 1443806239480,
  "active": true,
  "archived": false,
  "sizeOnDisk": 0
}
```

Related REST command: Copy a component version with all of its properties.

createAgentPool

Create an agent pool

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createAgentPool [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agents": [
    "Agent ID or name",
    "Agent ID or name (repeat as necessary)"
  ],
  "description": "Description",
  "name": "Name of the agent pool"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createAgentPool newPool.json
```

Example

Example JSON request

```
{
  "name": "New pool",
  "description": "My new agent pool",
  "agents": [
    "Agent1",
    "Agent2"
  ]
}
```

Example

Example response

```
{
  "id": "62eb25dd-edc0-4900-a4c7-25b440abb1a3",
  "name": "New pool",
  "active": true,
  "description": "My new agent pool",
  "status": "ONLINE"
}
```

Related REST command: Create an agent pool.

createApplication

Create an application from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createApplication [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Description",
  "enforceCompleteSnapshots": "Specify true to require an
  explicit version for each component",
  "name": "Application name or ID",
  "notificationScheme": "Notification scheme",
  "onlyChangedVersions": "If true, the 'Only Deploy
  Changed Versions' deployment option will be checked by
  default (default: true)",
  "teamMappings": [{
    "resourceRoleId": "ID of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
```

```
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "useWizard": "Specify true to use the Deployment Wizard
  when submitting application process request (default:
  false)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createApplication newApplication.json
```

Example

Example JSON request

```
{
  "name": "My new application",
  "description": "New application for command example",
  "notificationScheme": "Default Notification Scheme",
  "enforceCompleteSnapshots": "false"
}
```

Example

Example response

```
{
  "id": "5064e2b5-08aa-4e79-b0d5-9b63ba218d48",
  "name": "My new application",
  "description": "New application for command example",
  "created": 1390504790883,
  "enforceCompleteSnapshots": false,
  "active": true,
  "tags": [
  ],
  "user": "admin"
}
```

Related REST command: Create an application from a JSON file.

createApplicationFromTemplate

Create an application with components from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createApplicationFromTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Description",
  "enforceCompleteSnapshots": "Specify true to require an
  explicit version for each component",
  "existingComponentIds": "The list of existing Component
  IDs to attach the application.",
  "name": "Application name or ID",
  "notificationScheme": "Notification scheme",
  "teamMappings": [{
    "resourceRoleId": "ID of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "templateId": "The ID of the template to use. If you are
  using an application template, either this field or
  templateName are required.",
  "templateName": "The name of the template to use. If you
  are using an application template, either this field or
  templateId are required.",
  "templateProperties": [{
    "name": "The name of the property defined on the
  template. Required properties must be included here",
    "value": "Value of the property to set on the new
  application"
  }],
  "templateVersion": "The version of the template to use.
  This field is required if you are using an application
  template."
}
```

Related REST command: Create an application with components from a JSON file.

createApplicationPipeline

Create a new application pipeline

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createApplicationPipeline [parameters] [JSON file]
```

Parameters

Table 826. Parameters

Parameter	Type	Required	Description
application	string	true	Application name or id.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "name of the application pipeline",
  "stages": [{
    "blocks": [{
      "environment": "name of the environment for this
  block",
      "processes": ["process-name"]
    }],
    "name": "name of the stage"
  }]
}
```

Related REST command: Create a new application pipeline.

createApplicationProcess

Create an application process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createApplicationProcess [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Application ID",
  "description": "Description. This template includes one
  step, which deploys a component.",
  "disableSnapshots": "When true, prevent deployments
  using any snapshot and this application process",
  "inventoryManagementType": "The inventory management for
```

```
the process request, such as AUTOMATIC",
"name": "Application process name",
"offlineAgentHandling": "PRE_EXECUTION_CHECK",
"propDefs": [{
  "label": "Property label",
  "name": "Property name",
  "type": "TEXT"
}],
"properties": [{
  "description": "Description",
  "name": "Property name",
  "secure": true,
  "value": "Property value"
}],
"requiredRoleId": "Required Role Name or ID",
"rootActivity": {
  "children": [
    {
      "children": [{
        "children": [{
          "allowFailure": false,
          "children": [],
          "componentName": "Component name",
          "componentProcessName": "Component process
name",
          "name": "First step name",
          "properties": {},
          "type": "componentProcess"
        }],
        "componentName": "Component name",
        "name": "inventoryVersionCheck",
        "status": "Active",
        "type": "inventoryVersionDiff"
      }],
      "componentName": "Component name",
      "failFast": "false",
      "maxIteration": "-1",
      "name": "First step name",
      "runOnlyOnFirst": "false",
      "type": "componentEnvironmentIterator"
    },
    {
      "children": [],
      "name": "FINISH",
      "type": "finish"
    }
  ],
  "edges": [
    {
      "to": "First step name",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "from": "First step name",
      "to": "FINISH",
      "type": "ALWAYS",
      "value": ""
```

```
      }
    ],
    "name": "GRAPH",
    "offsets": [
      {
        "name": "First step name",
        "x": "-35",
        "y": "210"
      },
      {
        "name": "FINISH",
        "x": "0",
        "y": "420"
      }
    ],
    "propDefs": [],
    "type": "graph",
    "versionPresets": []
  }
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createApplication newApplicationProcess.json
```

Example

Example JSON request

To use an existing application process as a template, follow these steps:

1. Open the application process in the editor.
2. In your web browser, open the developer tools or install an extension that shows the network traffic from your browser to the server.
3. In the process editor, make a change to the application process. This change can be a trivial change, such as moving a step slightly. The process must need to be saved.
4. Save the application process.
5. In the web browser developer tools, find the REST PUT call to the endpoint `https://hostName:port/rest/deploy/applicationProcess/applicationProcessID/saveActivities`, where *`hostname`* and *`port`* are the host name and port of your server, and *`applicationProcessID`* is the ID of the application process. This PUT call is the call that sends the new version of the application process to the server.
6. Find the JSON payload of the REST PUT call. This code describes the steps in the process and the location of the steps on the editor. This payload starts with the code `{"type":"graph","layoutMode":`.
7. Copy the content of the JSON payload into the **rootActivity** property of your JSON request.
8. Update the other properties of the JSON request, such as **name** and **description**.

The following example request creates an application process that deploys a component.

```
{
  "name": "My new application process",
  "application": "79f33ffb-9d72-4967-9146-dc252eaea706",
  "description": "New application process for command example",
  "inventoryManagementType": "AUTOMATIC",
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
  "rootActivity": {
    "type": "graph",
    "name": "GRAPH",
    "edges": [
      {
        "to": "Deploy component",
        "type": "ALWAYS",
        "value": ""
      },
      {
        "to": "FINISH",
        "from": "Deploy component",
        "type": "ALWAYS",
        "value": ""
      }
    ],
    "offsets": [
      {
        "name": "Deploy component",
        "x": "-35",
        "y": "210"
      },
      {
        "name": "FINISH",
        "x": "0",
        "y": "420"
      }
    ],
    "children": [
      {
        "componentName": "Component Y",
        "failFast": "false",
        "runOnlyOnFirst": "false",
        "maxIteration": "-1",
        "type": "componentEnvironmentIterator",
        "name": "Deploy component",
        "children": [
          {
            "componentName": "Component Y",
            "status": "Active",
            "type": "inventoryVersionDiff",
            "name": "inventoryVersionCheck",
            "children": [
              {
                "componentName": "Component Y",
                "componentProcessName": "Deploy Component Y",
                "allowFailure": false,
                "properties": {
                },
                "type": "componentProcess",
                "name": "Deploy component",
                "children": [
```

```
                ]
              }
            ]
          }
        ]
      },
      {
        "type": "finish",
        "name": "FINISH",
        "children": [
        ]
      }
    ],
    "propDefs": [
    ]
  }
}
```

Example

Example response

```
{
  "id": "80d98207-3012-4d9f-858e-0b3aea2ce489",
  "name": "My new application process",
  "description": "New application process for command example",
  "active": true,
  "inventoryManagementType": "AUTOMATIC",
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
  "versionCount": 1,
  "version": 1,
  "commit": 0,
  "path": "applications/79f33ffb-9d72-4967-9146-dc252eaea706/processes/80d98207-3012-4d9f-858e-0b3aea2ce489"
}
```

Related REST command: Create an application process.

createApplicationTemplate

Create an application template from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createApplicationTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Application template description
  (Optional)",
```

```
  "enforceCompleteSnapshots": "Specify true to require an
  explicit version for each component",
  "name": "Application template name",
  "notificationScheme": "Notification scheme (Optional)",
  "onlyChangedVersions": "If true, the 'Only Deploy
  Changed Versions' deployment option will be checked by
  default (default: true)",
  "tagRequirements": [
    {
      "name": "Name of the Component tag",
      "number": "Number of components for the tag",
      "type": "GREATER_THAN, LESS_THAN, or EQUALS"
    },
    " (Optional)"
  ],
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    " (Optional)"
  ],
  "useWizard": "Specify true to use the Deployment Wizard
  when submitting application process request (default:
  false)"
}
```

Related REST command: Create an application template from a JSON file.

createAuthToken

Create an authentication token

Tokens provide authentication for agents, users, and external systems or applications.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createAuthToken [parameters]
```

Parameters

Table 827. Parameters

Parameter	Type	Required	Description
user	string	true	The user this token is associated with
expireDate	string	true	Expiration date and time of the token, in the format MM-DD-YYYY-HH:mm
description	string	false	Description of the authorization token
allowedIps	string	false	Allowed IP addresses in CIDR notation, such as 192.169.1.0/24
restriction	string	false	Name or ID of auth token restrictions to apply to created auth token. Required if set by administrator.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createAuthToken
  -user admin
  -expireDate 12-31-2020-12:00
```

Example

Example response

```
{"token":"42ece458-b197-4c2e-92a0-c63e10fd9b4f"}
```

Related REST command: Create an authentication token.

createAuthenticationRealm

Create an AuthenticationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createAuthenticationRealm [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "authorizationRealmIds": ["AuthorizationRealms"],
  "description": "A descrption (optional)",
  "name": "AuthenticationRealm name",
  "properties": {
    "auth-endpoint": "OIDC: The URI for authorization with
  the OpenID Provider",
    "authentication-mech": "LDAP: Simple or DIGEST-MD5",
    "client-id": "OIDC: The Client ID which is registered
  with the OpenID Provider",
    "client-secret": "OIDC: The Client Secret which is
  provided by the OpenID Provider upon registration",
    "connection-name": "LDAP: The username to use for
  Simple binding to LDAP for searches. If not specified, an
  anonymous connection is made.",
    "connection-password": "LDAP: The password to use for
  Simple binding to LDAP for searches.",
    "email-claim": "OIDC: The claim containing the user's
  email.",
    "end-session-endpoint": "OIDC: Optional: The URI for
  RP-initiated logout",
    "end-session-id-token-hint": "OIDC: If an ID token
  hint parameter (id_token_hint) is appended to the End
  Session Endpoint URL.",
    "groups-delim": "SSO: A string that acts as the
  delimiter between groups that are passed in the headers.
  For example, if the user should be added to groups GROUP1
  and GROUP2, and the header value was \"GROUP=GROUP1;
  GROUP=GROUP2\" the delimiter would be ';'. Note: special
  regex characters such as '+' must be escaped with one
  backslash ''.",
    "groups-header": "SSO: The header name that denotes
  the user groups to which user will be added.",
    "groups-regex": "SSO: The regular expression to find
  the groups in the header value. If a capturing group is
  specified, only the first will be caught (per delimited
  string). If no capturing group is specified with
  parentheses, then the whole delimited string will be
  captured.",
    "issuer": "OIDC: The issuer identifier for the issuer
  of the token response.",
    "jwks-uri": "OIDC: The JSON Web Key Sets URI endpoint",
    "name-claim": "OIDC: The claim containing user's name.",
    "sasl-realm": "LDAP: The sasl Realm for DIGEST-MD5
  authentication",
    "token-endpoint": "OIDC: 'The URI for token requests
  with the OpenID Provider",
    "url": "LDAP: The URL of the LDAP server. It should
  begin with 'ldap://' or 'ldaps://'. Failover servers can
  be added by separating the URLs with a space. Example:
  ldap://ldap.mydomain.com:389 ldap://ldap.mydomain2.com",
    "user-base": "LDAP: When you search multiple
  directories, specify the starting directory that is used
  for searches, such as ou=employees,dc=mydomain,dc=com.",
```

```
    "user-pattern": "LDAP: LDAP users exist in a single
  directory; use a pattern to create the DN for users.",
    "user-search": "LDAP: The LDAP filter expression to
  use when searching for user directory entries. The
  username is put in place of {0} in the search pattern. If
  this is an attribute and not part of the user DN, wrap in
  parentheses. E.g. uid={0} or (sAMAccountName={0})",
    "user-search-subtree": "LDAP: Search the full subtree
  for the user, as opposed to a single-level search only
  covering users directly inside the specified search base.
  \"true\" or \"false\"",
    "userinfo-endpoint": "OIDC: The URI for userinfo
  requests with the OpenID Provider"
  },
  "type": [
    "Any one of:",
    "INTERNAL",
    "LDAP",
    "SSO",
    "OIDC"
  ]
}
```

Related REST command: Create an AuthenticationRealm.

createAuthorizationRealm

Create an AuthorizationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createAuthorizationRealm [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "A descrption (optional)",
  "name": "AuthorizationRealm name",
  "properties": {
    "authentication-mech": "LDAP: Simple or DIGEST-MD5",
    "connection-name": "LDAP: The username to use for
  Simple binding to LDAP for searches. If not specified, an
  anonymous connection is made.",
    "connection-password": "LDAP: The password to use for
  Simple binding to LDAP for searches.",
    "group-attribute": "LDAP: The name of the attribute
  that contains group names in the user directory entry.",
    "group-base": "LDAP: The base directory to execute
  group searches in.",
    "group-name": "LDAP: The name of the entry that
  contains the user's group names in the directory entries
```

```
  returned by the group search. If this is not specified, no
  group search will take place.",
    "group-search": "LDAP: The LDAP filter expression to
  use when searching for group entries. The username will be
  put in place of {1} in the search pattern and the full
  user DN will be put in place of {0}.",
    "group-search-iterative": "LDAP: Perform additional
  lookups to find indirectly assigned groups. This option
  may significantly increase the amount of time required to
  authenticate users. Consult your LDAP administrators
  before enabling this option.",
    "group-search-subtree": "LDAP: Search the full subtree
  for the groups, as opposed to a single-level search only
  covering groups directly inside the specified search
  base.",
    "groups-delim": "SSO: A string that acts as the
  delimiter between groups that are passed in the headers.
  For example, if the user should be added to groups GROUP1
  and GROUP2, and the header value was \"GROUP=GROUP1;
  GROUP=GROUP2\" the delimiter would be ';'. Note: special
  regex characters such as '+' must be escaped with one
  backslash ''.",
    "groups-header": "SSO: The header name that denotes
  the user groups to which user will be added.",
    "groups-regex": "SSO: The regular expression to find
  the groups in the header value. If a capturing group is
  specified, only the first will be caught (per delimited
  string). If no capturing group is specified with
  parentheses, then the whole delimited string will be
  captured.",
    "sasl-realm": "LDAP: The sasl Realm for DIGEST-MD5
  authentication",
    "url": "LDAP: The URL of the LDAP server. It should
  begin with 'ldap://' or 'ldaps://'. Failover servers can
  be added by separating the URLs with a space. Example:
  ldap://ldap.mydomain.com:389 ldap://ldap.mydomain2.com",
    "user-base": "LDAP: When you search multiple
  directories, specify the starting directory that is used
  for searches, such as ou=employees,dc=mydomain,dc=com.",
    "user-pattern": "LDAP: LDAP users exist in a single
  directory; use a pattern to create the DN for users.",
    "user-search": "LDAP: The LDAP filter expression to
  use when searching for user directory entries. The
  username is put in place of {0} in the search pattern. If
  this is an attribute and not part of the user DN, wrap in
  parentheses. E.g. uid={0} or (sAMAccountName={0})",
    "user-search-subtree": "LDAP: Search the full subtree
  for the user, as opposed to a single-level search only
  covering users directly inside the specified search base.
  \"true\" or \"false\""
  },
  "type": [
    "Any one of:",
    "INTERNAL",
    "LDAP",
    "SSO"
  ]
}
```

Related REST command: Create an AuthorizationRealm.

createBlueprintDesignerIntegration

Create a blueprint designer integration from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createBlueprintDesignerIntegration [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Description",
  "name": "Blueprint Designer name or ID",
  "properties": {
    "landscaperPassword": "Landscaper password if
  useAdminCredentials is true",
    "landscaperUrl": "URL of Landscaper server",
    "landscaperUser": "Landscaper username if
  useAdminCredentials is true",
    "useAdminCredentials": "Boolean value true/false"
  }
}
```

Related REST command: Create a blueprint designer integration from a JSON file.

createComponent

Create a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createComponent [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "cleanupCountToKeep": "Number of most recent versions to
  keep; omit this parameter to inherit the system settings
  (Optional)",
  "cleanupDaysToKeep": "Number of days to keep versions;
  omit this parameter to inherit the system settings
```

```
  (Optional)",
  "defaultVersionType": "The default type to create
  versions with; valid values are FULL and INCREMENTAL",
  "description": "Description",
  "importAutomatically": "Specify true to import new
  versions automatically",
  "inheritSystemCleanup": "Inherit the system cleanup
  settings. If set to true, this overrides the other cleanup
  attributes. (Optional, Boolean)",
  "integrationAgent": "ID of the Agent to use for
  importing new component versions (Optional)",
  "integrationTag": "Tag name of the Agent to use for
  importing new component versions (Optional)",
  "name": "Component name or ID",
  "properties": {"Source configuration plugin property
  name": "Property value"},
  "sourceConfigPlugin": "The name of the source
  configuration plug-in to use (Optional)",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "templateId": "ID of the template to use (Optional)",
  "templateName": "Name of a template to use; this
  parameter is ignored if  templateId is specified
  (Optional)",
  "templateVersion": "Version number of the specified
  template (Optional)",
  "useVfs": "Boolean"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createComponent newComponent.json
```

Example

Example JSON requests

```
{
  "name": "My new component",
  "description": "New component for command example",
  "importAutomatically": false,
  "useVfs": true,
  "sourceConfigPlugin": "File System (Versioned)",
```

```
  "defaultVersionType": "FULL",
  "properties": {
    "FileSystemVersionedComponentProperties\/basePath":
    "/opt/newcomponent"
  }
}
```

The **properties** attribute in the JSON template refers to properties for the source configuration plug-in, not custom properties. For example, the following code creates a component from a Maven repository. It includes properties that are prefixed with `MavenComponentProperties` to refer to the properties for that plug-in.

```
{
  "name": "Component from Maven",
  "description": "New component for command example for Maven",
  "importAutomatically": false,
  "useVfs": false,
  "sourceConfigPlugin": "Maven",
  "defaultVersionType": "FULL",
  "properties": {
    "MavenComponentProperties\/artifactId":"MyArtifact",
    "MavenComponentProperties\/groupId":"MyGroup",
    "MavenComponentProperties\/extension":"ext",
    "MavenComponentProperties\/repoUrl":"http:\/\/myserver.com"
  }
}
```

To see the properties for a source configuration plug-in, create a component in the web interface and use the source configuration plug-in. Then, then export its information with the getComponent on page 2751 command. The JSON that this command returns includes the properties for the source configuration plug-in. You can also look in the `plugin.xml` file in the plug-in.

To create a component from a template, specify either the **templateId** attribute or the **templateName** attribute. Then, specify the **templateVersion** attribute; to use the most recent version of the template, specify an empty string. If the template contains property definitions, you can specify values for those properties with the attribute **`template/`***`propertyName`*, where *`propertyName`* is the name of the property. For example, the following JSON code creates a component from the most recent version of the **Template1** template. This code specifies the value `MyValue` for the parameter **template/MyProperty**.

```
{
  "name":"NewComponent1",
  "description":"",
  "templateName":"Template1",
  "templateVersion":"",
  "template/MyProperty":"MyValue",
  "componentType":"STANDARD",
  "sourceConfigPlugin":"",
  "importAutomatically":"false",
  "useVfs":"true",
  "defaultVersionType":"FULL",
  "importAgentType":"inherit",
  "inheritSystemCleanup":"true",
  "runVersionCreationProcess":"false",
  "properties":{},
  "teamMappings":[]
}
```

Example

Example response

```
{
  "id": "e84c7947-fad1-4399-8eb8-0d0d354b9e2b",
  "name": "My new component",
  "description": "New component for command example",
  "created": 1391456765556,
  "importAutomatically": false,
  "useVfs": true,
  "active": true,
  "deleted": false,
  "defaultVersionType": "FULL",
  "cleanupDaysToKeep": 0,
  "cleanupCountToKeep": 0,
  "tags": [
  ],
  "user": "admin"
}
```

Related REST command: Create a component.

createComponentProcess

Create a new component process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createComponentProcess [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "active": "true",
  "component": "Component ID",
  "configActionType": "ADD",
  "defaultWorkingDir": "${p:resource/work.dir}/${p:
  component.name}",
  "description": "Description. This template includes two
  steps: Download Artifacts and Run Shell as a good starting
  point for your component process.",
  "inventoryActionType": "ADD",
  "name": "Name of the component process",
  "propDefs": [],
  "rootActivity": {
    "children": [
      {
        "allowFailure": false,
        "children": [],
```

```
      "commandName": "Download Artifacts",
      "impersonationUseSudo": false,
      "name": "First step",
      "pluginName": "Versioned File Storage",
      "pluginVersion": 16,
      "properties": {
        "artifactSetBaseDir": "",
        "compId": "${p:component.id}",
        "directoryOffset": ".",
        "fileExcludePatterns": "",
        "fileIncludePatterns": "**/*",
        "fullVerification": true,
        "label": "${p:version.name",
        "repositoryId": "${p:
component/code_station/repository}",
        "repositoryUrl": "${p:server.url}/vfs",
        "resId": "${p:resource.id}",
        "serverUrl": "${p:server.url}",
        "setFileExecuteBits": false,
        "syncMode": true,
        "verifyFileIntegrity": false
      },
      "showHidden": false,
      "type": "plugin",
      "useImpersonation": false
    },
    {
      "allowFailure": false,
      "children": [],
      "commandName": "Shell",
      "impersonationUseSudo": false,
      "name": "Second step",
      "pluginName": "Shell",
      "pluginVersion": 3,
      "properties": {
        "directoryOffset": ".",
        "outputFile": "",
        "runAsDaemon": false,
        "scriptBody": "echo \"Hello World\"",
        "shellInterpreter": ""
      },
      "showHidden": false,
      "type": "plugin",
      "useImpersonation": false
    },
    {
      "children": [],
      "name": "FINISH",
      "type": "finish"
    }
  ],
  "edges": [
    {
      "to": "First step",
      "type": "ALWAYS",
      "value": ""
    },
    {
```

```
        "from": "First step",
        "to": "Second step",
        "type": "ALWAYS",
        "value": ""
      },
      {
        "from": "Second step",
        "to": "FINISH",
        "type": "ALWAYS",
        "value": ""
      }
    ],
    "layoutMode": "auto",
    "name": "GRAPH",
    "offsets": [
      {
        "name": "First step",
        "x": "-65",
        "y": "90"
      },
      {
        "name": "Second step",
        "x": "-70",
        "y": "210"
      },
      {
        "name": "FINISH",
        "x": "-60",
        "y": "450"
      }
    ],
    "type": "graph"
  },
  "status": "Active",
  "takesVersion": "Specify true for a deployment or
  rollback process; specify false for an operational (no
  version needed) process"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createComponentProcess newComponentProcess.json
```

Example

Example JSON request

```
{
  "name": "My new component process",
  "component": "e84c7947-fad1-4399-8eb8-0d0d354b9e2b"
  "description": "New component process for command example",
  "defaultWorkingDir": "${p:resource\/work.dir}\/${p:component.name}",
  "takesVersion": "false",
  "inventoryActionType": "ADD",
```

```
"status": "Active",
"configActionType": "ADD",
"active": "true",
"propDefs": [
],
"rootActivity": {
  "type": "graph",
  "name": "GRAPH",
  "edges": [
    {
      "to": "First Step",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "to": "Second Step",
      "from": "First Step",
      "type": "ALWAYS",
      "value": ""
    },
    {
      "to": "FINISH",
      "from": "Second Step",
      "type": "ALWAYS",
      "value": ""
    }
  ],
  "offsets": [
    {
      "name": "First Step",
      "x": "-65",
      "y": "90"
    },
    {
      "name": "Second Step",
      "x": "-70",
      "y": "210"
    },
    {
      "name": "FINISH",
      "x": "-60",
      "y": "450"
    }
  ],
  "children": [
    {
      "allowFailure": false,
      "useImpersonation": false,
      "showHIdden": false,
      "impersonationUseSudo": false,
      "commandName": "Download Artifacts",
      "pluginName": "DevOps Deploy Versioned File Storage",
      "pluginVersion": 18,
      "type": "plugin",
      "name": "First Step",
      "children": [
      ],
      "properties": {
```

```
          "directoryOffset": ".",
          "artifactSetBaseDir": "",
          "fileIncludePatterns": "**\/*",
          "fileExcludePatterns": "",
          "syncMode": true,
          "fullVerification": true,
          "setFileExecuteBits": false,
          "verifyFileIntegrity": false,
          "repositoryUrl": "${p:server.url}\/vfs",
          "repositoryId": "${p:component\/code_station\/repository}",
          "label": "${p:version.name",
          "serverUrl": "${p:server.url}",
          "compId": "${p:component.id}",
          "resId": "${p:resource.id}"
        }
      },
      {
        "allowFailure": false,
        "useImpersonation": false,
        "showHIdden": false,
        "impersonationUseSudo": false,
        "commandName": "Shell",
        "pluginName": "Shell",
        "pluginVersion": 3,
        "type": "plugin",
        "name": "Second Step",
        "children": [
        ],
        "properties": {
          "directoryOffset": ".",
          "shellInterpreter": "",
          "scriptBody": "echo \"Hello World\"",
          "runAsDaemon": false,
          "outputFile": ""
        }
      },
      {
        "type": "finish",
        "name": "FINISH",
        "children": [
        ]
      }
    ]
  }
}
```


Example

Example response


```
{
  "id": "76f0d90a-b9c5-42d2-a561-6ab1b22a1acb",
  "name": "My new component process",
  "description": "New component process for command example",
  "defaultWorkingDir": "${p:resource/work.dir}/${p:component.name}",
  "takesVersion": false,
  "inventoryActionType": "ADD",
  "status": "Active",
```

```
  "active": true,
  "versionCount": 1,
  "version": 1,
  "commit": 0,
  "path": "components/bd486abe-e6e1-43e5-9e0f-f7ad20929bad/processes/76f0d90a-b9c5-42d2-a561-6ab1b22a1acb"
}
```

Related REST command: Create a new component process.

createComponentTemplate

Create a component template from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createComponentTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "componentType": "STANDARD or ZOS",
  "description": "Component template description",
  "importAutomatically": "Optional, boolean. Force
  components to import versions automatically (true) or
  never import automatically (false). Leave blank to specify
  value when component is created.",
  "integrationAgent": "ID of the Agent to use for
  importing new component versions (Optional)",
  "integrationTag": "Tag name of the Agent to use for
  importing new component versions (Optional)",
  "name": "Component template name or ID",
  "properties": {"Source configuration plugin property
  name": "Property value"},
  "sourceConfigPluginName": "Name of the source
  configuration type"
}
```

Related REST command: Create a component template from a JSON file.

createDeploymentTrigger

Create a deployment trigger.

This command creates a deployment trigger for the given environment, component, application process, and user.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createDeploymentTrigger [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationProcess": "Required. ID of the application
  process.",
  "component": "Required. Name or ID of the component.",
  "environment": "Required. ID of the environment.",
  "executingUser": "Required. Name or ID of the user to
  execute the deployment."
}
```

Related REST command: Create a deployment trigger..

createEnvironment

Create a new environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createEnvironment [parameters] [JSON file]
```

Parameters

Table 828. Parameters

Parameter	Type	Required	Description
application	string	false	Application to add the environment to
name	string	false	Name of the new environment
description	string	false	Description of the new environment
color	string	false	Color of the new environment, in hex format. For a REST call, URL encode the # symbol; for example, %23ff0000 for pure red. For a udclient command, put the color in hex format in quotes; for example '#ff0000'.
requireApprovals	boolean	false	Whether the environment requires approvals

Table 828. Parameters (continued)

Parameter	Type	Required	Description
noSelfApprovals	boolean	false	Whether the environment allows self approvals

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationId": "Application ID associated with the
  Environment template.",
  "cleanupCountToKeep": "Number of most recently deployed
  versions to keep (optional)",
  "cleanupDaysToKeep": "Number of days to keep versions
  deployed to this environment (optional)",
  "color": "HTML color code for the environment (optional)",
  "description": "Description (optional)",
  "exemptProcessesArray": "An array of process Ids. These
  processes will be exempt from approvals for this
  environment (optional).",
  "externalApprovalAgent": "Name or id of agent to run
  external approval processes on (optional).",
  "externalApprovalAgentPool": "Name or id of agent pool
  to run external approval processes on (optional).",
  "externalApprovalProcess": "Name or id of external
  approval process(optional).",
  "historyCleanupDaysToKeep": "Number of days to keep
  application process history for this environment
  (optional)",
  "inheritSystemCleanup": "Inherit the system cleanup
  settings. If set to true, this overrides the other cleanup
  attributes. (optional, boolean)",
  "lockSnapshots": "Whether snapshots will be locked when
  deployed to this environment (optional)",
  "name": "Environment name",
  "noSelfApprovals": "When this option is selected, users
  that submit deployment requests cannot approve their own
  requests (optional)",
  "requireApprovals": "Whether approvals will be required
  (optional)",
  "requireSnapshot": "If enabled, all deployments must use
  a snapshot instead of loose versions (optional)",
  "snapshotDaysToKeep": "Number of days to keep snapshots
  deployed to this environment (optional)",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
```

```
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "useSystemDefaultDays": "Whether or not to use the
  system default number of days to keep application process
  history for this environment (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createEnvironment
  -application JPetStore
  -name NewEnvironment
  -color '#ff0000'
```

Related REST command: Create a new environment.

createEnvironmentFromTemplate

Create a new environment from template

This command creates and Environment from a template and attaches it to the specified application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createEnvironmentFromTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationId": "Application ID associated with the
  Environment template.",
  "description": "Description (optional)",
  "name": "Environment name",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
```

```
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "templateId": "ID of the template to use, either this
  field or templateName are required.",
  "templateName": "Name of a template to use, either this
  field or templateId are required.",
  "templateProperties": [{
    "name": "The name of the property defined on the
  template. Required properties must be included here",
    "value": "Value of the property to set on the new
  environment"
  }]
}
```

Related REST command: Create a new environment from template.

createEnvironmentTemplate

Create a new environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createEnvironmentTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationTemplateId": "ID of the application template
  to which the environment template belongs",
  "cleanupCountToKeep": "Number of most recently deployed
  versions to keep (Optional)",
  "cleanupDaysToKeep": "Number of days to keep versions
  deployed to this environment (Optional)",
  "color": "HTML color code for the environment template",
  "description": "description of the template (Optional)",
  "environmentType": "type of environment, leave blank for
  Standard Environment",
  "exemptProcessesArray": "An array of process Ids. These
  processes will be exempt from approvals for this
  environment (Optional).",
  "externalApprovalAgent": "Name or id of agent to run
  external approval processes on (Optional).",
  "externalApprovalAgentPool": "Name or id of agent pool
  to run external approval processes on (Optional).",
  "externalApprovalProcess": "Name or id of external
  approval process(Optional).",
  "historyCleanupDaysToKeep": "Number of days to keep
  application process history for this environment
```

```
  (Optional)",
  "lockSnapshots": "Whether snapshots will be locked when
  deployed to this environment (Optional)",
  "name": "name of the environment template",
  "noSelfApprovals": "When this option is selected, users
  that submit deployment requests cannot approve their own
  requests (Optional)",
  "requireApprovals": "Whether approvals will be required
  (Optional)",
  "requireSnapshot": "If enabled, all deployments must use
  a snapshot instead of loose versions (Optional)",
  "resourceTemplate": "ID of the resource template to use",
  "snapshotDaysToKeep": "Number of days to keep snapshots
  deployed to this environment (Optional)",
  "snapshotLockType": "type of snapshot lock: ALL,
  VERSIONS, CONFIGURATION (Required with \"lockSnapshots\"",
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (Optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (Optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ],
  "useSystemDefaultDays": "Whether or not to use the
  system default number of days to keep application process
  history for this environment (Optional)"
}
```

Related REST command: Create a new environment template.

createGenericProcess

Create a generic process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createGenericProcess [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "defaultResourceId": "DEPRECATED, legacy format which is
```

```
single default resource ID.",
"defaultResourceIds": ["JSONArray of Default resource ID
(s) This property takes precedence when both
defaultResourceIds and defaultResourceId are present"],
"description": "Description. This template includes two
steps: Download Artifacts and Run Shell as a good starting
point for your process.",
"name": "Process name or ID",
"notificationScheme": "Name of notification scheme",
"propDefs": [{
  "allowedValues": [{"label": "value"}],
  "defaultLabel": "defaultLabel",
  "description": "description",
  "httpBasePath": "httpBasePath",
  "httpFormat": "httpFormat",
  "httpLabelPath": "httpLabelPath",
  "httpPassword": "httpPassword",
  "httpUrl": "httpurl",
  "httpUsername": "httpUsername",
  "httpValuePath": "httpValuePath",
  "label": "label",
  "name": "name",
  "pattern": "Pattern",
  "required": "required",
  "resolveHttpValuesUrl": "resolveHttpValuesUrl",
  "type": "type",
  "value": "value"
}],
"properties": {"Property name": "Property value
(Optional)"},
"rootActivity": {
  "children": [
    {
      "allowFailure": false,
      "children": [],
      "commandName": "Download Artifacts",
      "impersonationUseSudo": false,
      "name": "First step",
      "pluginName": "Versioned File Storage",
      "pluginVersion": 16,
      "properties": {
        "artifactSetBaseDir": "",
        "compId": "${p:component.id}",
        "directoryOffset": ".",
        "fileExcludePatterns": "",
        "fileIncludePatterns": "**/*",
        "fullVerification": true,
        "label": "${p:version.name",
        "repositoryId": "${p:
component/code_station/repository}",
        "repositoryUrl": "${p:server.url}/vfs",
        "resId": "${p:resource.id}",
        "serverUrl": "${p:server.url}",
        "setFileExecuteBits": false,
        "syncMode": true,
        "verifyFileIntegrity": false
      },
      "showHidden": false,
```

```
        "type": "plugin",
        "useImpersonation": false
      },
      {
        "allowFailure": false,
        "children": [],
        "commandName": "Shell",
        "impersonationUseSudo": false,
        "name": "Second step",
        "pluginName": "Shell",
        "pluginVersion": 3,
        "properties": {
          "directoryOffset": ".",
          "outputFile": "",
          "runAsDaemon": false,
          "scriptBody": "echo \"Hello World\"",
          "shellInterpreter": ""
        },
        "showHidden": false,
        "type": "plugin",
        "useImpersonation": false
      },
      {
        "children": [],
        "name": "FINISH",
        "type": "finish"
      }
    ],
    "edges": [
      {
        "to": "First step",
        "type": "ALWAYS",
        "value": ""
      },
      {
        "from": "First step",
        "to": "Second step",
        "type": "ALWAYS",
        "value": ""
      },
      {
        "from": "Second step",
        "to": "FINISH",
        "type": "ALWAYS",
        "value": ""
      }
    ],
    "name": "GRAPH",
    "offsets": [
      {
        "name": "First step",
        "x": "-65",
        "y": "90"
      },
      {
        "name": "Second step",
        "x": "-70",
        "y": "210"
```

```
      },
      {
        "name": "FINISH",
        "x": "-60",
        "y": "450"
      }
    ],
    "type": "graph"
  },
  "workingDir": "${p:resource/work.dir}/${p:process.name}"
}
```

Related REST command: Create a generic process.

createGroup

Create an empty group

Groups can only be created in Internal Storage authorization realms.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createGroup [parameters]
```

Parameters

Table 829. Parameters

Parameter	Type	Required	Description
group	string	true	Name of the group
description	string	false	Description of the group
authorizationRealm	string	true	Name of the Internal Storage authorization realm

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createGroup
  -group group1
  -authorizationRealm "Internal Security"
```

Note: Adding groups to LDAP and SSO realms is deprecated.

Example

Example response

```
{
  "id": "a5be904f-d65a-41ae-9bfd-0a0384c90eeb",
  "name": "group1",
  "enabled": true
}
```

Related REST command: Create an empty group.

createMaintenanceBlackout

Create a maintenance blackout for the given date range. The date range cannot overlap with any existing maintenance blackout date ranges.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createMaintenanceBlackout [parameters]
```

Parameters

Table 830. Parameters

Parameter	Type	Required	Description
startDate	string	true	The date the maintenance blackout starts. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time.
endDate	string	true	The date the maintenance blackout ends. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time.
name	string	false	(Optional) The name or description of the blackout.
recurrence	string	false	(Optional) A recurrence character for repeating blackouts (D - daily, W - weekly, M - monthly)

Related REST command: Create a maintenance blackout for the given date range.

createManyVersions

Create multiple component versions

Do not use the keywords latest, latestVersion, or newest as the version name.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createManyVersions [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"wrappedArray": [{
  "componentId": "Component identifier. If not specified
  componentName is required",
  "componentName": "Component name. If not specified
  componentId is required",
  "links": [{
    "isPriority": "Optional. If true, prioritize the link
  by placing it in the version page header for quicker
  access",
    "link": "Link url",
    "name": "Link name"
  }],
  "status": [{"name": "Status name"}],
  "version": "Version name to create"
}]}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createManyVersions
  newVersions.json
```

Example

Example JSON request

```
[
  {
    "version": "New version 1",
    "componentName": "MyComponent",
    "links": [
      {
        "name": "New link",
        "link": "http://www.foo.com"
      }
    ],
    "status": [
      {
        "name": "MyStatus"
      }
    ]
  },
  {
    "version": "New version 2",
    "componentName": "MyComponent",
    "links": [
      {
```

```
        "name": "New link",
        "link": "http://www.foo.com"
      }
    ],
    "status": [
      {
        "name": "MyStatus"
      }
    ]
  }
]
```

Related REST command: Create multiple component versions.

createPostProcessingScript

Create a post processing script from a JSON file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createPostProcessingScript [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "body": "The body of the script (required)",
  "description": "The description of the script (optional)",
  "name": "The name of the script (required)",
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ]
}
```

Related REST command: Create a post processing script from a JSON file.

createResource

Create a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createResource [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agent": "Name or ID of an agent to use for automation
  (Optional)",
  "agentPool": "Name or ID of an agent pool to use for
  automation (Optional)",
  "component": "Name or ID of a component to associate
  with the resource (Optional)",
  "componentTag": "Name or ID of a component tag to
  associate with the resource; any components with that tag
  can be deployed to this resource (Optional)",
  "description": "Description",
  "impersonationForce": "Specify true to prevent children
  or steps from specifying their own impersonation settings
  (Optional)",
  "impersonationGroup": "Group for default impersonation
  (Optional)",
  "impersonationPassword": "Password for default
  impersonation. Agents running on Linux and Unix machines
  will ignore this value (Optional)",
  "impersonationUseSudo": "Specify true to use the sudo
  command for impersonation. Agents running on Windows
  machines will ignore this value (Optional)",
  "impersonationUser": "User name for default
  impersonation (Optional)",
  "inheritTeam": "Whether or not to set inherit teams for
  resource. teamMappings will be ingored if true. Defaults
  to true (Optional)",
  "name": "Name for the new resource; leave blank to
  inherit the name from an agent, pool, or role",
  "parent": "ID or path to parent resource; leave blank to
  create a root resource (Optional)",
  "role": "Name or ID of a resource role to associate with
  the agent; if this resource is for a component, specify
  the component name instead (Optional)",
  "roleProperties": {"Property name": "Property value
  (Optional)"},
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
```

```
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createResource newResource.json
```

Example

Example JSON requests

This JSON request creates a new top-level resource:

```
{
  "name": "My new resource",
  "description": "New resource for command example"
}
```

This JSON request creates a resource group within an existing resource tree:

```
{
  "name": "New child resource",
  "description": "New resource for command example",
  "parent": "/TopLevelResource/subgroup"
}
```

This JSON request creates a component resource. In this case, the parent resource must be an agent resource.

```
{
    "name":"ComponentA",
    "role":"ComponentA",
    "description": "Description of ComponentA",
    "roleProperties": {"MyProp":"Value1"},
    "parent": "/TopLevelResource/agent1",
    "impersonationUser":"user1",
    "impersonationGroup":"group1",
    "impersonationPassword":"password",
    "impersonationUseSudo":false,
    "impersonationForce":true

}
```

This JSON request creates an agent resource. In this case, the **agentID** parameter is the ID of an agent.

```
{
  "agentId": "0ac81f92-aeb9-34ce-a6c5-93b6aed380f5",
  "name": "agent1",
  "parent": "/agents"
}
```

Example

Example response

```
{
  "id": "6a2b106b-ce03-44cd-88db-b89de202013c",
  "name": "My new resource",
  "path": "/My new resource",
  "active": true,
  "description": "New resource for command example",
  "inheritTeam": false,
  "impersonationPassword": "****",
  "impersonationUseSudo": false,
  "impersonationForce": false,
  "hasAgent": false,
  "tags": [
  ]
}
```

Related REST command: Create a resource.

createResourceTemplate

Create a resource template from a JSON file

This method supports importing a resource template from a cloud system.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createResourceTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "cloudResourceId": "ID of cloud resource; leave blank
  when not cloud-based template (Optional)",
  "connection": "ID of cloud connection; leave blank when
  not cloud-based template (Optional)",
  "description": "Description",
  "isVsys": "Specify 'true' when cloud resource is Virtual
  System Pattern (non-classic) in IPAS 2.0 or ICO 2.0 cloud
  product, otherwise specify 'false' when cloud resource is
  using Weaver (for IWD, SCO, ICO, and IPAS versions before
  2.0 and 'classic' virtual system pattern in IPAS 2.0 or
  ICO 2.0); leave blank when not cloud-based template
  (Optional)",
  "name": "Name for the resource template",
  "parentId": "ID of parent resource template; leave blank
  when none or creating cloud-based template (Optional)",
  "rootTeamMappings": "Array of team ids; if not
```

```
  specified, no teams will be mapped to Root Resource;
  format: [ {teamId: '<ID-of-team-1>'}, {teamId: '<ID-of-
  team-2>'}, ... ]; (Optional)",
  "teamMappings": "Array of team ids; if not specified, no
  teams will be mapped to Template; format: [ {teamId: '<ID-
  of-team-1>'}, {teamId: '<ID-of-team-2>'}, ... ];
  (Optional)"
}
```

Related REST command: Create a resource template from a JSON file.

createResourceTemplateWithAgentPrototypes

Create a resource template with agent prototypes from a JSON file

This method supports creating a resource template with multiple agent prototypes

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createResourceTemplateWithAgentPrototypes [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agentPrototypes": [{
    "agentNamePatterns": "List of name patterns (one per
  line) for matching agents; (Optional)",
    "componentTags": "List of component tags to be mapped
  to the prototype; format: [ {componentTag: '<name-or-id-of-
  comp-tag-1>'}, {componentTag: '<name-or-id-of-comp-tag-
  2>'}, ... ]; (Optional)",
    "components": "List of components to be mapped to the
  prototype; format: [ {component: '<name-or-ID-of-comp-
  1>'}, {component: '<name-or-ID-of-comp-2>'}, ... ];
  (Optional)",
    "description": "Description",
    "name": "Name for the agent prototype",
    "teamMappings": "Array of team ids; if not specified,
  no teams will be mapped to Template; format: [ {teamId:
  '<ID-of-team-1>'}, {teamId: '<ID-of-team-2>'}, ... ];
  (Optional)"
  }],
  "description": "Description",
  "name": "Name for the resource template",
  "teamMappings": "Array of team ids; if not specified, no
  teams will be mapped to Template; format: [ {teamId: '<ID-
  of-team-1>'}, {teamId: '<ID-of-team-2>'}, ... ];
  (Optional)"
}
```

Related REST command: Create a resource template with agent prototypes from a JSON file.

createSnapshot

Create a snapshot

This command creates a snapshot by specifying the components (optional, by IDs and/or names) for the snapshot, rather than taking a snapshot of an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createSnapshot [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Application name or ID",
  "description": "Description",
  "name": "Snapshot name",
  "versions": "JSON array of mappings from components by
  name or ID to versions by name or ID, such as [{\"
  JPetStore-APP\": \"latest\"}, {\"JPetStore-DB\": \"1.0\"},
  {\"3d6cd2fd-3381-4b95-b851-1aa7f83e225e\":\"c2c61215-c20a-
  4bc9-8f34-7f9f41de385b\"}]"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createSnapshot newSnapshot.json
```

Example

Example JSON request

```
{
  "name": "My snapshot",
  "application": "JPetStore",
  "description": "JPetStore snapshot",
  "versions": [
    {"JPetStore-APP": "1.1"},
    {"JPetStore-DB": "1.0"},
    {"JPetStore-WEB": "1.1"}
  ]
}
```

Example

Example JSON request

You can also use the keywords `latest`, `latestVersion`, and `newest` in place of component version names. In this case, the process uses the most recently created component version. The following example shows how you can use these keywords in the JSON request.

```
{
  "name": "My snapshot",
  "application": "JPetStore",
  "description": "Snapshots of most recently created component versions",
  "versions": [
    {"JPetStore-APP": "latest"},
    {"JPetStore-DB": "latestVersion"},
    {"JPetStore-WEB": "newest"}
  ]
}
```

Example

Example response

```
{
  "id": "ff9274d4-bc3d-493c-90aa-5020e2fda56d",
  "name": "My snapshot",
  "description": "JPetStore snapshot",
  "created": 1391451659050,
  "active": true,
  "locked": false
}
```

Related REST command: Create a snapshot.

createSnapshotOfEnvironment

Take a snapshot of an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createSnapshotOfEnvironment [parameters]
```

Parameters

Table 831. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID

Table 831. Parameters (continued)

Parameter	Type	Required	Description
name	string	true	Name of the new snapshot
description	string	false	Snapshot description
excludeUnmapped	boolean	false	Exclude components with no resource in this environment (default false)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createSnapshotOfEnvironment
  -environment newEnvironment
  -application JPetStore
  -name NewSnapshot
```

Example

Example response

```
{
  "id": "41eab076-deb6-4746-8b2d-9f491103a06c",
  "name": "NewSnapshot"
}
```

Related REST command: Take a snapshot of an environment.

createStatus

Create a status

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createStatus [parameters]
```

Parameters

Table 832. Parameters

Parameter	Type	Required	Description
type	string	true	Type of status (inventory, version, or snapshot)
status	string	true	Name of the new status

Table 832. Parameters (continued)

Parameter	Type	Required	Description
color	string	false	Color of the new status (HTML color strings or valid color names) (Optional)
description	string	false	Description of the new status (Optional)
unique	string	false	Specify true to make this status unique (Optional)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createStatus
  -type version
  -status Documented
  -color "#0077FF"
```

Example

Example response

```
{
  "id": "abff7bb6-bf95-417e-bf00-b0a8678e832a",
  "type": "VERSION",
  "name": "Documented",
  "active": true,
  "color": "#0077FF",
  "unique": false
}
```

Related REST command: Create a status.

createTeam

Create a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createTeam [parameters]
```

Parameters

Table 833. Parameters

Parameter	Type	Required	Description
team	string	true	Name of the team

Table 833. Parameters (continued)

Parameter	Type	Required	Description
description	string	false	Description of the team

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createTeam
  -team MyNewTeam
  -description "MyNewTeamDescription"
```

Example

Example response

```
{
  "id": "769c2486-ed40-4a3a-ac52-faa33d30cfd5",
  "name": "MyNewTeam",
  "description": "MyNewTeamDescription",
  "isDeletable": true
}
```

Related REST command: Create a team.

createUser

Create a user

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createUser [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "authenticationRealm": "Authentication realm ID",
  "email": "Email address",
  "name": "User name",
  "password": "Password"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createUser
  newUser.json
```

Example

Example JSON request

```
{
  "name": "jsmith",
  "password": "password",
  "actualName": "Jane Smith",
  "email": "jsmith@example.org",
  "authenticationRealm": "Internal Security"
}
```

Example

Example response

```
{
  "id": "2507eaaf-c98e-46ee-936d-bc538bee9dc4",
  "name": "jsmith",
  "actualName": "Jane Smith",
  "displayName": "Jane Smith (jsmith)",
  "email": "jsmith@example.org",
  "deleted": false,
  "deletedDate": 0,
  "authenticationRealm": "20000000000000000000000000000001",
  "isLockedOut": false,
  "isDeletable": true
}
```

Related REST command: Create a user.

createVaultAppRole

Create a new Vault App Role

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createVaultAppRole [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "The Vault App Role Name (Required, Unique per
```

```
  Vault store)",
  "roleId": "The Vault Role ID (Required)",
  "secretId": "The Vault Secret ID (Optional)",
  "store": "Name/ID of the Vault store (Required)"
}
```

Related REST command: Create a new Vault App Role.

createVaultCertificate

Create a new Vault Certificate

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createVaultCertificate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "certificatePEM": "The certificate to be stored in PEM
  format(Required)",
  "name": "The Vault Certificate Name (Required, Unique
  per Vault store)",
  "privateKeyPEM": "The private key to be stored in PEM
  format(Required)",
  "store": "Name/ID of the Vault store (Required)"
}
```

Related REST command: Create a new Vault Certificate.

createVaultLdapUser

Create a new Vault Ldap User

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createVaultLdapUser [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "The Vault Ldap User Name (Required, Unique per
  Vault store)",
```

```
  "password": "The Vault Ldap Users password (Optional)",
  "store": "Name/ID of the Vault store (Required)"
}
```

Related REST command: Create a new Vault Ldap User.

createVaultSecretStore

Create a secret store.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createVaultSecretStore [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Secret Store description",
  "name": "Secret Store name",
  "namespace": "Vault Server Namespace (optional)",
  "teamMappings": {
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  },
  "type": "vault",
  "url": "Vault Server URL"
}
```

Related REST command: Create a secret store..

createVersion

Create a component version

Do not use the keywords latest, latestVersion, or newest as the version name.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createVersion [parameters]
```

Parameters

Table 834. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the new version
description	string	false	Description of the new version
type	string	false	Type to use for the new version (incremental or full) - full is the default.
sourceConfigExecutionRecordId	string	false	A system parameter. This should not be provided by the user or by user scripts.
importing	boolean	false	Whether or not to mark this version as importing. Version marked as importing cannot be used in deployments. Default false

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  createVersion
  -component JPetStore-APP
  -name 1.2
```

Example

Example response

```
{
  "id": "c0198aaf-6f4b-4d2e-b103-a7f6a257f580",
  "name": "1.2",
  "type": "FULL",
  "created": 1390572847946,
  "active": true,
  "archived": false
}
```

Related REST command: Create a component version.

createWebhookTemplate

Create a new webhook template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  createWebhookTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "body": "The webhook template body specified with Apache
  Velocity template syntax",
  "description": "Description (Optional)",
  "name": "Name of the new webhook template",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the webhook
  template to, either this field or teamId are needed to
  specify a team"
  }],
  "type": "Webhook template type [PROCESS_SUCCESS,
  PROCESS_FAILURE, PROCESS_STARTED, APPROVAL_COMPLETED,
  APPROVAL_FAILED, EXTERNAL_APPROVAL_COMPLETED,
  EXTERNAL_APPROVAL_FAILED, SCHEDULED_PROCESS_NOT_STARTED,
  TASK_CREATED]"
}
```

Related REST command: Create a new webhook template.

deleteAgent

Delete an agent

Agents should be shut down before deletion.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAgent [parameters]
```

Parameters

Table 835. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteAgent
  -agent Agent1
```

Related REST command: Delete an agent.

deleteAgentPool

Delete an agent pool

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAgentPool [parameters]
```

Parameters

Table 836. Parameters

Parameter	Type	Required	Description
pool	string	true	Name or ID of the agent pool

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteAgentPool
  -pool Pool1
```

Related REST command: Delete an agent pool.

deleteAgentProperty

Remove a property from an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAgentProperty [parameters]
```

Parameters

Table 837. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteAgentProperty
  -agent Agent1
  -name Prop1
```

Related REST command: Remove a property from an agent.

deleteAllDesiredInventoryVersions

Delete all versions for a component in an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAllDesiredInventoryVersions [parameters]
```

Parameters

Table 838. Parameters

Parameter	Type	Required	Description
deleteVersions	boolean	false	If set to true, versions will be removed from the environment's inventory.
retainDays	int	false	If set to greater than 0, inventory will only be removed if created more than this many days ago.
environmentId	string	true	
componentId	string	true	Name or ID of the component

Related REST command: Delete all versions for a component in an environment.

deleteApplication

Delete an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteApplication [parameters]
```

Parameters

Table 839. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Related REST command: Delete an application.

deleteApplicationPipeline

Delete an application pipeline

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteApplicationPipeline [parameters]
```

Parameters

Table 840. Parameters

Parameter	Type	Required	Description
application	string	false	Application name or id. Required if pipeline specified by name.
pipeline	string	true	Pipeline name or id.

Related REST command: Delete an application pipeline.

deleteApplicationProcess

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteApplicationProcess [parameters]
```

Parameters

Table 841. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
processName	string	true	Name of the application process

Related REST command: Delete an application process.

deleteApplicationTemplate

Delete an application template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteApplicationTemplate [parameters]
```

Parameters

Table 842. Parameters

Parameter	Type	Required	Description
applicationTemplate	string	true	Name or ID of the application

Related REST command: Delete an application template.

deleteAuthenticationRealm

Delete an AuthenticationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAuthenticationRealm
```

Parameters

Table 843. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the AuthenticationRealm

Related REST command: Delete an AuthenticationRealm.

deleteAuthorizationRealm

Delete an AuthorizationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteAuthorizationRealm [parameters]
```

Parameters

Table 844. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the authorization realm

Related REST command: Delete an AuthorizationRealm.

deleteBlueprintDesignerIntegration

Delete a blueprint designer integration

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteBlueprintDesignerIntegration [parameters]
```

Parameters

Table 845. Parameters

Parameter	Type	Required	Description
patternIntegration	string	true	UUID or Name of the Blueprint Pattern Integration

Related REST command: Delete a blueprint designer integration.

deleteComponent

Delete a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteComponent [parameters]
```

Parameters

Table 846. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Related REST command: Delete a component.

deleteComponentVersionPropDef

Delete a component version property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteComponentVersionPropDef [parameters]
```

Parameters

Table 847. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the property to delete

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteComponentVersionPropDef
  -component JPetStore-APP
  -name Prop1
```

Related REST command: Delete a component version property.

deleteDeploymentTrigger

Delete a deployment trigger.

This command deletes a deployment trigger with the given ID.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteDeploymentTrigger
```

Parameters

Table 848. Parameters

Parameter	Type	Required	Description
deploymentTriggerId	string	true	ID of the deployment trigger.

Related REST command: Delete a deployment trigger..

deleteEnvironment

Delete an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteEnvironment [parameters]
```

Parameters

Table 849. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
deleteAttachedResources	boolean	false	Specify true to delete all resources in this environment, including all subresources. If this value and 'deleteCloudInstances' is set to true, then all agent objects in use by this environment will be deleted when the agent goes offline. (Optional)

Table 849. Parameters (continued)

Parameter	Type	Required	Description
deleteCloudInstances	boolean	false	Specify true to delete any associated instances in the cloud provider; this parameter is valid only for environments that were created dynamically along with the environment (Optional)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteEnvironment
  -application JPetStore
  -environment NewEnvironment
```

Related REST command: Delete an environment.

deleteEnvironmentGate

Delete the environment gate of a given environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteEnvironmentGate [parameters]
```

Parameters

Table 850. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of environment
application	string	false	Name or ID of application (required if environment is specified by name)

Related REST command: Delete the environment gate of a given environment.

deleteEnvironmentTemplate

Delete an environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteEnvironmentTemplate [parameters]
```

Parameters

Table 851. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	ID of the environment template

Related REST command: Delete an environment template.

deleteEnvironmentTemplateGate

Delete the environment gate of a given environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteEnvironmentTemplateGate [parameters]
```

Parameters

Table 852. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	Name or ID of environment template
applicationTemplate	string	false	Name or ID of application template (required if environment template is specified by name)

Related REST command: Delete the environment gate of a given environment template.

deleteFrontendServer

Delete a frontend server.

Used to remove frontend servers which are no longer active.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteFrontendServer
```

Parameters

Table 853. Parameters

Parameter	Type	Required	Description
serverId	string	true	Unique id of the server from installed.properties.

Related REST command: Delete a frontend server..

deleteGroup

Delete a group

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteGroup [parameters]
```

Parameters

Table 854. Parameters

Parameter	Type	Required	Description
group	string	true	Name of the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteGroup
  -group group1
```

Related REST command: Delete a group.

deleteJavaPackage

Delete a java package.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteJavaPackage [parameters]
```

Parameters

Table 855. Parameters

Parameter	Type	Required	Description
package	string	true	Name or ID of the java package

Related REST command: Delete a java package..

deleteMaintenanceBlackout

Cancel the given maintenance blackout. If it is a real event (synthetic=false) in a recurring, only the individual event is canceled. Synthetic events cannot be cancelled with this API.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteMaintenanceBlackout
```

Parameters

Table 856. Parameters

Parameter	Type	Required	Description
id	string	true	The id of the maintenance blackout to cancel.

Related REST command: Cancel the given maintenance blackout.

deletePostProcessingScript

Delete a post processing script.

Used to remove processing scripts.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deletePostProcessingScript [parameters]
```

Parameters

Table 857. Parameters

Parameter	Type	Required	Description
script	string	true	Name/ID of the post processing script.

Related REST command: Delete a post processing script..

deleteRedundantDesiredInventoryVersions

Delete all redundant versions for a component in an environment

A version is considered redundant if all of its files have been replaced by versions that have been deployed more recently to the same environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteRedundantDesiredInventoryVersions [parameters]
```

Parameters

Table 858. Parameters

Parameter	Type	Required	Description
deleteRedundant	boolean	false	If set to true, redundant versions will be removed from the environment's inventory.
retainDays	int	false	If set to greater than 0, inventory will only be removed if created more than this many days ago.
environmentId	string	true	
componentId	string	true	Name or ID of the component

Related REST command: Delete all redundant versions for a component in an environment.

deleteResource

Delete a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteResource [parameters]
```

Parameters

Table 859. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Example

Example

```
udclient  -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteResource
  -resource "/ResourceGroup/My resource"
```

Note: When you specify the name of the resource, type the complete path to the resource, including the initial '/' character if the resource is not in any higher-level group.

Related REST command: Delete a resource.

deleteResourceProperty

Delete a resource property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteResourceProperty [parameters]
```

Parameters

Table 860. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)

Table 860. Parameters (continued)

Parameter	Type	Required	Description
application	string	false	Name or ID of the application (Required to look up blueprint by name)
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteResourceProperty
  -resource "/ResourceGroup/My resource"
  -name property1
```

Note: When you specify the name of the resource, type the complete path to the resource, including the initial '/' character if the resource is not in any higher-level group.

Related REST command: Delete a resource property.

deleteResourceRoleProperty

Delete a resource role property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteResourceRoleProperty [parameters]
```

Parameters

Table 861. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
name	string	true	Name of the property

Related REST command: Delete a resource role property.

deleteSecretStore

Delete a secret store.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteSecretStore [parameters]
```

Parameters

Table 862. Parameters

Parameter	Type	Required	Description
store	string	true	Name or id of the secret store.

Related REST command: Delete a secret store..

deleteServerKeystoreEntry

Delete a server keystore entry

Deletes an entry from the server keystore which contains key pairs and certificates used for agent communication. This keystore is the file at path appdata/conf/server.keystore. A 409 Conflict status is returned if the alias no longer corresponds to the same certificate at the time of the request. If the alias was never present or was removed concurrently, this operation reports success.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteServerKeystoreEntry [parameters]
```

Parameters

Table 863. Parameters

Parameter	Type	Required	Description
alias	string	true	The alias that will be removed.
certificateFingerprint	string	true	The certificate fingerprint that must match to check for concurrent modification.

Related REST command: Delete a server keystore entry.

deleteSnapshot

Delete a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteSnapshot [parameters]
```

Parameters

Table 864. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteSnapshot
  -application MyApplication
  -snapshot Snapshot1
```

Related REST command: Delete a snapshot.

deleteStatus

Delete a status

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteStatus [parameters]
```

Parameters

Table 865. Parameters

Parameter	Type	Required	Description
type	string	false	Type of status (inventory, version, or snapshot); this parameter is required you specify the name of the status instead of the ID
status	string	true	Name or ID of the status

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteStatus
  -type version
  -status Tested
```

Related REST command: Delete a status.

deleteTeam

Delete a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteTeam [parameters]
```

Parameters

Table 866. Parameters

Parameter	Type	Required	Description
team	string	true	Name of the team

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteTeam
  -team MyNewTeam
```

Related REST command: Delete a team.

deleteUser

Delete a user

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteUser [parameters]
```

Parameters

Table 867. Parameters

Parameter	Type	Required	Description
user	string	true	Name/ID of the user. The name will be case-sensitive if a case-sensitive realm is provided.
realm	string	false	Limit any user name search to given AuthenticationRealm.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteUser
  -user jsmith
```

Related REST command: Delete a user.

deleteVaultAppRole

Delete a Vault AppRole

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteVaultAppRole [parameters]
```

Parameters

Table 868. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultAppRole	string	true	Name or ID of the Vault App Role

Related REST command: Delete a Vault AppRole.

deleteVaultCertificate

Delete a Vault Certificate

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteVaultCertificate [parameters]
```

Parameters

Table 869. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultCertificate	string	true	Name or ID of the Vault Certificate

Related REST command: Delete a Vault Certificate.

deleteVaultLdapUser

Delete a Vault Ldap User

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteVaultLdapUser [parameters]
```

Parameters

Table 870. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Store ID/Name
vaultLdapUser	string	true	Name or ID of the Vault Ldap User

Related REST command: Delete a Vault Ldap User.

deleteVersion

Delete a version from a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteVersion [parameters]
```

Parameters

Table 871. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component
version	string	true	Name or ID of the version(s) to be deleted. This option can be repeated to delete multiple versions.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  deleteVersion
  -component MyComponent
  -version 1.0
```

Related REST command: Delete a version from a component.

deleteWebhookTemplate

Delete a webhook template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  deleteWebhookTemplate [parameters]
```

Parameters

Table 872. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	Name or ID of the webhook template to remove

Related REST command: Delete a webhook template.

downloadJavaPackage

Download available Java package.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  downloadJavaPackage [parameters]
```

Parameters

Table 873. Parameters

Parameter	Type	Required	Description
package	string	true	Name or ID of the java package

Related REST command: Download available Java package..

downloadVersionArtifacts

Download the artifacts of a component version

This command exports the artifacts of a component version into an archive file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  downloadVersionArtifacts [parameters]
```

Parameters

Table 874. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version
location	string	true	Path to download the artifacts to. You must have permission to create files in this location.
singleFilePath	string	false	Optional path to an individual version artifact to download. If specified, only that file will be downloaded.
encoding	string	false	Optional character encoding to determine the file names of the artifacts. Defaults to the encoding of the file system.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  downloadVersionArtifacts
  -component JPetStore-WEB
  -version 1.0
  -location .
```

Related REST command: Download the artifacts of a component version.

executeApplicationPipeline

Execute an application pipeline

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  executeApplicationPipeline [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Application name or id. Only required
  when pipeline is referenced by name.",
  "pipeline": "Pipeline name or id",
  "snapshot": "Name or id of snapshot. Only snapshot or
  versions should be specified.",
  "versions": [{
    "component": "Name or id of component. Required when
  version referenced by name",
    "version": "Version name or id."
  }]
}
```

Related REST command: Execute an application pipeline.

exportApplication

Export an application to a json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportApplication [parameters]
```

Parameters

Table 875. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Related REST command: Export an application to a json file..

exportApplicationProcess

Export an application process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportApplicationProcess [parameters]
```

Parameters

Table 876. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
applicationProcess	string	true	Name of the process

Example

Command to export an application process in JSON format

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  exportApplicationProcess
  -application JPetStore
  -applicationProcess "Deploy JPetStore"
```

Example

Response

```
File written to: ./application-process-Deploy JPetStore.json
```

Related REST command: Export an application process.

exportComponent

Export a component to a json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportComponent [parameters]
```

Parameters

Table 877. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Related REST command: Export a component to a json file..

exportComponentProcess

Export a component process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportComponentProcess [parameters]
```

Parameters

Table 878. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
componentProcess	string	true	Name or ID of the component process

Related REST command: Export a component process.

exportComponentTemplate

Export a component template to a json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportComponentTemplate [parameters]
```

Parameters

Table 879. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template

Example

Command to export a component template process in YAML format

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  exportComponentTemplateProcess
  -componentTemplate JPetStoreTemplate
  -componentProcess "Deploy JPetStore Comp"
```

Example

Response

```
File written to: ./component-process-Deploy JPetStore Comp.yml
```

Related REST command: Export a component template to a json file..

exportComponentTemplateProcess

Export a component template process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportComponentTemplateProcess [parameters]
```

Parameters

Table 880. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template
componentProcess	string	true	Name or ID of the component process

Related REST command: Export a component template process.

exportDetailedGroup

List the members of a group (detailed)

This command lists the members of a group, including IDs and e-mail addresses.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportDetailedGroup [parameters]
```

Parameters

Table 881. Parameters

Parameter	Type	Required	Description
group	string	true	Name of the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  exportDetailedGroup
  -group teamA
```

Example

Example response

```
[
  {
    "userName": "jdoe",
    "userId": "a459dffd-e057-494d-b119-16e85d7cd911",
    "userEmail": "jdoe@example.org"
  },
  {
    "userName": "jsmith",
    "userId": "7794dee7-d4c6-4f6f-b1ce-56c5d3cbfdda",
    "userEmail": "jsmith@example.org"
  }
]
```

Related REST command: List the members of a group (detailed).

exportGenericProcess

Export a process to a json file

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportGenericProcess [parameters]
```

Parameters

Table 882. Parameters

Parameter	Type	Required	Description
process	string	true	Name or ID of the generic process

Related REST command: Export a process to a json file.

exportGroup

List the members of a group

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportGroup [parameters]
```

Parameters

Table 883. Parameters

Parameter	Type	Required	Description
group	string	true	Name of the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  exportGroup
  -group group1
```

Example

Example response

```
{
  "name": "group1",
  "users": [
    "jsmith",
    "jdoe"
```

```
  ]
}
```

Related REST command: List the members of a group.

exportSnapshot

Export a snapshot to a zip file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  exportSnapshot [parameters]
```

Parameters

Table 884. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Related REST command: Export a snapshot to a zip file..

finishedImporting

Mark a version as completed importing

Mark a version as completed importing so it is available for use in deployments

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  finishedImporting [parameters]
```

Parameters

Table 885. Parameters

Parameter	Type	Required	Description
version	string	true	Name or id of version
component	string	false	Name or id of the component. This is required if version name is used

Related REST command: Mark a version as completed importing.

forceCancelWorkflow

Force cancel a workflow in progress

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  forceCancelWorkflow [parameters]
```

Parameters

Table 886. Parameters

Parameter	Type	Required	Description
workflow	string	true	ID of the workflow to force cancel.

Related REST command: Force cancel a workflow in progress.

forceRenameAgent

Rename an agent

Rename an agent. Use this command if running the agent configuration script in the agent bin directory is not convenient. Only the administrator can run this method.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  forceRenameAgent [parameters]
```

Parameters

Table 887. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent to rename
name	string	true	The new name for the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  forceRenameAgent
```

```
  -agent Agent1
  -name NewAgentName
```

Related REST command: Rename an agent.

forgetAgentCert

Forget Agent Certificates

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  forgetAgentCert [parameters]
```

Parameters

Table 888. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent. This can be repeated for multiple agents.

Related REST command: Forget Agent Certificates.

getAgent

Get information about an agent

This command returns a JSON representation of an agent.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgent [parameters]
```

Parameters

Table 889. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
```

```
  getAgent
  -agent Agent1
```

Example

Example response


```
{
  "id": "18cc6e9c-0c18-44bc-b34d-53c8609d9c44",
  "securityResourceId": "4c9f52c0-6f90-4e6a-94a9-5a68c8081796",
  "name": "myAgent",
  "active": true,
  "licensed": false,
  "licenseType": "NONE",
  "status": "ONLINE",
  "version": "8.0.0.0.123456",
  "lastContact": 1498229460223,
  "dateCreated": 1497277343513,
  "workingDirectory": "/opt/devops-deploy/agent/var/work/",
  "impersonationPassword": "****",
  "impersonationUseSudo": false,
  "impersonationForce": false,
  "tags": [],
  "security": {
    "read": true,
    "execute": true,
    "Add to Agent Pool": true,
    "Create Resources": true,
    "Delete": true,
    "Edit Basic Settings": true,
    "Execute on Agents": true,
    "Manage Impersonation": true,
    "Manage Properties": true,
    "Manage Teams": true,
    "Manage Version Imports": true,
    "View Agents": true
  },
  "propSheet": {
    "id": "42763bdd-66b5-4021-9a57-62ba3dd745fd",
    "path": "agents\/18cc6e9c-0c18-44bc-b34d-53c8609d9c44\/propSheet",
    "version": 2,
    "versionCount": 2,
    "commit": 55,
    "versioned": true
  },
  "extendedSecurity": {
    "read": true,
    "execute": true,
    "Add to Agent Pool": true,
    "Create Resources": true,
    "Delete": true,
    "Edit Basic Settings": true,
    "Execute on Agents": true,
    "Manage Impersonation": true,
    "Manage Properties": true,
    "Manage Teams": true,
    "Manage Version Imports": true,
    "View Agents": true,
```

```
    "teams": [{
      "teamId": "8213c4ec-1f66-4b4a-9a73-e24bf8b8102f",
      "teamLabel": "My team",
      "resourceTypeId": "20000000000000000000000000000106",
      "resourceTypeName": "Agent"
    }]
  }
}
```

Related REST command: Get information about an agent.

getAgentJavaHomes

Get list of managed Java installations

This command returns a list of managed Java installations for an agent.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentJavaHomes [parameters]
```

Parameters

Table 890. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Related REST command: Get list of managed Java installations.

getAgentMaintenanceTask

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentMaintenanceTask [parameters]
```

Parameters

Table 891. Parameters

Parameter	Type	Required	Description
task	string	true	ID of maintenance task

Related REST command: Get agent maintenace task.

getAgentMaintenanceTasks

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentMaintenanceTasks [parameters]
```

Parameters

Table 892. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Related REST command: Get agent maintenace tasks.

getAgentPool

Get information about an agent pool

Returns a JSON representation of an agent pool.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentPool [parameters]
```

Parameters

Table 893. Parameters

Parameter	Type	Required	Description
pool	string	true	Name or ID of the agent pool

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAgentPool
  -pool Pool1
```

Example

Example response

```
{
  "id": "8bfa2e62-2376-466c-91c5-3f79b3e6f655",
  "name": "Pool1",
  "active": true,
  "description": "",
```

```
  "status": "ONLINE",
  "security": {
    "read": true,
    "write": true
  },
  "agentIds": "897fa851-fab1-455f-bd84-fe83ae43c441,bdc2019c-78ae-403c-a528-22b31ae3c183",
  "extendedSecurity": {
    "read": true,
    "write": true,
    "Create Agent Pools": true,
    "Edit Agent Pools": true,
    "View Agent Pools": true,
    "teams": [
      {
        "teamId": "ea307e2e-d6be-4092-a7e8-e8e02274de7e",
        "teamLabel": "My team"
      }
    ]
  }
}
```

Related REST command: Get information about an agent pool.

getAgentPools

Get information about all agent pools

Return a JSON representation of all agent pools.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentPools [parameters]
```

Parameters

Table 894. Parameters

Parameter	Type	Required	Description
active	boolean	false	

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAgentPools
```

Example

Example response

```
[
  {
    "id": "11dda768-525d-4055-a7d2-5fb0e86ad236",
    "name": "Pool1",
    "active": true,
    "description": "",
    "status": "ONLINE"
  },
  {
    "id": "8bfa2e62-2376-466c-91c5-3f79b3e6f655",
    "name": "Pool2",
    "active": true,
    "description": "",
    "status": "ONLINE"
  }
]
```

Related REST command: Get information about all agent pools.

getAgentProperty

Get the value of a property on an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentProperty [parameters]
```

Parameters

Table 895. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAgentProperty
  -agent Agent1
  -name JAVA_HOME
```

Related REST command: Get the value of a property on an agent.

getAgentPrototypes

Get all agent prototypes with this resource template as their ancestor

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentPrototypes [parameters]
```

Parameters

Table 896. Parameters

Parameter	Type	Required	Description
template	string	true	Name or ID of resource template

Related REST command: Get all agent prototypes with this resource template as their ancestor.

getAgentStatuses

Returns a JSONObject with count of agents to status. Expected format: { "ONLINE":4, "UPGRADE_RECOMMENDED":1 }
Requires Settings Tab permission

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgentStatuses
```

Parameters

None.

Related REST command: Get agent statuses.

getAgents

Get information about all agents

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAgents [parameters]
```

Parameters

Table 897. Parameters

Parameter	Type	Required	Description
active	boolean	false	

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAgents
```

Example

Example response

```
[
  {
    "id": "bdc2019c-78ae-403c-a528-22b31ae3c183",
    "name": "Agent1",
    "active": true,
    "licensed": false,
    "licenseType": "NONE",
    "status": "ONLINE",
    "version": "6.0.1.0.452394",
    "workingDirectory": "/opt/devops-deploy/agent/var/work/",
    "impersonationPassword": "****",
    "impersonationForce": false,
    "tags": [
    ]
  }
]
```

Related REST command: Get information about all agents.

getAllApplicationProcessesForApplication

Get all application processes of an application

Returns a JSON representation of all application processes for a specific application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllApplicationProcessesForApplication [parameters]
```

Parameters

Table 898. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Related REST command: Get all application processes of an application.

getAllComponentProcessRequestCount

Get all component process requests submitted after a certain time

This command returns a count of all the component process requests submitted after a certain time.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllComponentProcessRequestCount [parameters]
```

Parameters

Table 899. Parameters

Parameter	Type	Required	Description
submittedAfter	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server counts all CPRs that were created after the specified timestamp.
includeGhosted	boolean	false	True or false, whether to include ghosted components in the result.

Related REST command: Get all component process requests submitted after a certain time.

getAllComponentsForTemplate

Get a list of all components using this template

This command returns a JSON representation of a list of components.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllComponentsForTemplate [parameters]
```

Parameters

Table 900. Parameters

Parameter	Type	Required	Description
template	string	true	Name or ID of the component template

Related REST command: Get a list of all components using this template.

getAllLatestDesiredInventory

Get all of the latest desired inventory for an environment.

This command returns a JSON array of all of the desired inventory for an environment. It optionally groups the desired inventory by component.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllLatestDesiredInventory
```

Parameters

Table 901. Parameters

Parameter	Type	Required	Description
environmentId	string	true	Name or ID of the environment
groupVersions	boolean	true	True if the versions should be grouped by component.

Related REST command: Get all of the latest desired inventory for an environment..

getAllProcesses

Get information on all generic processes

This command returns a JSON representation of all generic processes.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllProcesses
```

Parameters

None.

Related REST command: Get information on all generic processes.

getAllTeams

List teams

This command lists all teams on the server, all users in each team, and all of the roles that each user has on each team.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllTeams
```

Parameters

None.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAllTeams
```

Example

Example response


```
[
  {
    "id": "20000000000000000000000100000000",
    "name": "System Team",
    "isDeletable": true,
    "roleMappings": [
      {
        "user": {
          "id": "20000000000000000000000001000000",
          "name": "admin",
          "displayName": "admin",
          "deleted": false,
          "deletedDate": 0,
          "authenticationRealm": "20000000000000000000000000000001",
          "isLockedOut": false,
          "isDeletable": true
        },
        "role": {
          "id": "20000000000000000000000000010001",
          "name": "Administrator",
          "description": "",
          "isDeletable": true
        }
      }
    ],
    "applications": [],
    "environments": [],
    "processes": []
  },
  {
    "id": "e7b24877-049a-4c52-b2a1-c5c5a78895bd",
```

```
    "name": "TeamA",
    "isDeletable": true,
    "roleMappings": [
      {
        "user": {
          "id": "a459dffd-e057-494d-b119-16e85d7cd911",
          "name": "jdoe",
          "actualName": "Jane Doe",
          "displayName": "Jane Doe (jdoe)",
          "email": "jdoe@example.org",
          "deleted": false,
          "deletedDate": 0,
          "authenticationRealm": "20000000000000000000000000000001",
          "isLockedOut": false,
          "isDeletable": true
        },
        "role": {
          "id": "3b76daf9-460b-4a43-ae70-d750219c6569",
          "name": "Developer",
          "description": "",
          "isDeletable": true
        }
      },
      {
        "user": {
          "id": "7794dee7-d4c6-4f6f-b1ce-56c5d3cbfdda",
          "name": "jsmith",
          "actualName": "Joe Smith",
          "displayName": "Joe Smith (jsmith)",
          "email": "jsmith@example.org",
          "deleted": false,
          "deletedDate": 0,
          "authenticationRealm": "20000000000000000000000000000001",
          "isLockedOut": false,
          "isDeletable": true
        },
        "role": {
          "id": "3b76daf9-460b-4a43-ae70-d750219c6569",
          "name": "Developer",
          "description": "",
          "isDeletable": true
        }
      },
      {
        "user": {
          "id": "7794dee7-d4c6-4f6f-b1ce-56c5d3cbfdda",
          "name": "jsmith",
          "actualName": "Joe Smith",
          "displayName": "Joe Smith (jsmith)",
          "email": "jsmith@example.org",
          "deleted": false,
          "deletedDate": 0,
          "authenticationRealm": "20000000000000000000000000000001",
          "isLockedOut": false,
          "isDeletable": true
        },
        "role": {
          "id": "20000000000000000000000000010001",
```

```
        "name": "Administrator",
        "description": "",
        "isDeletable": true
      }
    }
  ],
  "applications": [],
  "environments": [],
  "processes": []
},
{
  "id": "fe036470-0bfd-46d8-9529-11be16597435",
  "name": "TeamB",
  "isDeletable": true,
  "roleMappings": [
    {
      "user": {
        "id": "a459dffd-e057-494d-b119-16e85d7cd911",
        "name": "jdoe",
        "actualName": "Jane Doe",
        "displayName": "Jane Doe (jdoe)",
        "email": "jdoe@example.org",
        "deleted": false,
        "deletedDate": 0,
        "authenticationRealm": "20000000000000000000000000000001",
        "isLockedOut": false,
        "isDeletable": true
      },
      "role": {
        "id": "3b76daf9-460b-4a43-ae70-d750219c6569",
        "name": "Developer",
        "description": "",
        "isDeletable": true
      }
    },
    {
      "user": {
        "id": "7794dee7-d4c6-4f6f-b1ce-56c5d3cbfdda",
        "name": "jsmith",
        "actualName": "Joe Smith",
        "displayName": "Joe Smith (jsmith)",
        "email": "jsmith@example.org",
        "deleted": false,
        "deletedDate": 0,
        "authenticationRealm": "20000000000000000000000000000001",
        "isLockedOut": false,
        "isDeletable": true
      },
      "role": {
        "id": "3b76daf9-460b-4a43-ae70-d750219c6569",
        "name": "Developer",
        "description": "",
        "isDeletable": true
      }
    },
    {
      "user": {
        "id": "a459dffd-e057-494d-b119-16e85d7cd911",
```

```
          "name": "jdoe",
          "actualName": "Jane Doe",
          "displayName": "Jane Doe (jdoe)",
          "email": "jdoe@example.org",
          "deleted": false,
          "deletedDate": 0,
          "authenticationRealm": "20000000000000000000000000000001",
          "isLockedOut": false,
          "isDeletable": true
        },
        "role": {
          "id": "20000000000000000000000000010001",
          "name": "Administrator",
          "description": "",
          "isDeletable": true
        }
      }
    ],
    "applications": [],
    "environments": [],
    "processes": []
  }
]
```


Related REST command: List teams.

getAllUsersInRoleForTeam

List all users with a given role on a team

This command returns all users in a role for a given team, including users that are associated via a group membership

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllUsersInRoleForTeam [parameters]
```

Parameters

Table 902. Parameters

Parameter	Type	Required	Description
role	string	true	Name or id of the role
team	string	true	Name or id of the team

Related REST command: List all users with a given role on a team.

getAllUsersWithEmail

Get all users with a given e-mail address

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAllUsersWithEmail [parameters]
```

Parameters

Table 903. Parameters

Parameter	Type	Required	Description
email	string	true	E-mail address of the user

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  allUsersWithEmail
  -email jsmith@us.ibm.com
```

Example

Example response

```
[{"id":"bade154b-c8c9-4aa6-a875-4b56d7132be5","name":"jsmith"},
{"id":"2c1c9012-2111-4d6a-b5f0-b6cabff94713","name":"janesmith"}]
```

Related REST command: Get all users with a given e-mail address.

getApplication

Get information about an application

This command returns a JSON representation of an application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplication [parameters]
```

Parameters

Table 904. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplication
  -application JPetStore
```

Example

Example response

```
{
  "id": "c8a45972-da39-4450-87f0-21fae710e0f5",
  "name": "JPetStore",
  "description": "",
  "created": 1387378155981,
  "enforceCompleteSnapshots": false,
  "active": true,
  "tags": [
  ],
  "user": "admin",
  "componentCount": 3,
  "extendedSecurity": {
    "read": true,
    "write": true,
    "Create Applications": true,
    "Edit Applications": true,
    "Manage Snapshots": true,
    "Run Component Processes": true,
    "View Applications": true,
    "teams": [
    ]
  }
}
```

Related REST command: Get information about an application.

getApplicationPipeline

Get an application pipeline definition

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationPipeline [parameters]
```

Parameters

Table 905. Parameters

Parameter	Type	Required	Description
application	string	false	Application name or id. Required if pipeline specified by name.
pipeline	string	true	Pipeline name or id.

Related REST command: Get an application pipeline definition.

getApplicationPipelineExecution

Get an application pipeline execution

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationPipelineExecution [parameters]
```

Parameters

Table 906. Parameters

Parameter	Type	Required	Description
executionId	string	false	Id of the pipeline execution.

Related REST command: Get an application pipeline execution.

getApplicationPipelineExecutions

Get a list of up to 100 application pipeline executions

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationPipelineExecutions [parameters]
```

Parameters

Table 907. Parameters

Parameter	Type	Required	Description
application	string	false	Application name or id. Required if pipeline argument is a name.

Table 907. Parameters (continued)

Parameter	Type	Required	Description
pipeline	string	false	Pipeline name or id.
startIndex	int	false	Index of the first result to return. At most 100 results will be returned.
dateRangeBegin	long	false	Specifies the date range beginning in miiliseconds since epoch.
dateRangeEnd	long	false	Specifies the date range end in miiliseconds since epoch.

Related REST command: Get a list of up to 100 application pipeline executions.

getApplicationPipelines

Get a list of up to 100 application pipelines

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationPipelines [parameters]
```

Parameters

Table 908. Parameters

Parameter	Type	Required	Description
application	string	true	Application name or id.
startIndex	int	false	Index of first pipeline to be returned.

Related REST command: Get a list of up to 100 application pipelines.

getApplicationProcess

Get information about an application process

Returns a JSON representation of an application process.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcess [parameters]
```

Parameters

Table 909. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
applicationProcess	string	true	Name of the process

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProcess
  -application JPetStore
  -applicationProcess "Deploy JPetStore"
```

Example

Example response

```
{
  "id": "b02d0960-e878-45b5-88d9-7461159f0e19",
  "name": "Deploy JPetStore",
  "description": "",
  "active": true,
  "inventoryManagementType": "AUTOMATIC",
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
  "versionCount": 2,
  "version": 2,
  "commit": 30,
  "path": "applications/c8a45972-da39-4450-87f0-21fae710e0f5/processes/b02d0960-e878-45b5-88d9-7461159f0e19"
}
```

Related REST command: Get information about an application process.

getApplicationProcessExecution

Get an application process request's activity trace.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessExecution
```

Parameters

Table 910. Parameters

Parameter	Type	Required	Description
request	string	true	The ID of the application process request.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProcessExecution
  -request 151cd268-ef32-4e55-903b-86ee66314e1e
```

Example

Example response


```
{
  "id": "8cbccccf-581e-4500-af41-1d0429ef2dae",
  "type": "graph",
  "name": "2da0c9e5-0045-45b3-a49e-6f130afa0305",
  "state": "EXECUTING",
  "result": "NONE",
  "startDate": 1441394363997,
  "duration": 9052,
  "paused": false,
  "workflowTraceId": "193e892f-677d-4534-9763-ff8d005a5829",
  "children": [
    {
      "id": "bcbc874b-bbd9-4c6c-8890-3575373bc07f",
      "type": "runProcess",
      "displayName": "Run generic process",
      "name": "MyGenericProcess",
      "state": "EXECUTING",
      "result": "NONE",
      "startDate": 1441394363997,
      "duration": 9052,
      "workflowTraceId": "193e892f-677d-4534-9763-ff8d005a5829",
      "childRequestId": "94c77d8d-e6d2-41fd-aa8d-61f606c438e5",
      "graphPosition": 1
    },
    {
      "id": "a9ef585b-ff08-4b25-bd4d-fe7295a8882e",
      "type": "componentEnvironmentIterator",
      "displayName": "Install Component 2",
      "name": "3f13d0d7d892af1b135251d94583a9",
      "state": "INITIALIZED",
      "result": "NONE",
      "duration": 1441394373049,
      "workflowTraceId": "193e892f-677d-4534-9763-ff8d005a5829",
```

```
      "component": {
        "id": "92c2b418-b3f1-4a5b-bfb7-a88ff5d6c268",
        "securityResourceId": "63a4fe6e-1a8a-4d1e-8122-4f0abca2faa4",
        "name": "Component 2",
        "description": "",
        "created": 1441381387613,
        "componentType": "STANDARD",
        "ignoreQualifiers": 0,
        "importAutomatically": false,
        "useVfs": true,
        "active": true,
        "integrationFailed": false,
        "deleted": false,
        "defaultVersionType": "FULL",
        "cleanupDaysToKeep": 0,
        "cleanupCountToKeep": 0,
        "tags": [],
        "user": "admin"
      },
      "graphPosition": 2
    },
    {
      "id": "326a0b17-360a-43cf-9645-7f58264db996",
      "type": "componentEnvironmentIterator",
      "displayName": "Install Component 1",
      "name": "8eb1ec1009597252ce10c89e2b948b",
      "state": "INITIALIZED",
      "result": "NONE",
      "duration": 1441394373137,
      "workflowTraceId": "193e892f-677d-4534-9763-ff8d005a5829",
      "component": {
        "id": "23742526-8105-4146-bcf1-e4fce1734805",
        "securityResourceId": "9e4cca84-83f2-42ed-b970-3003fef92dd5",
        "name": "Component 1",
        "description": "",
        "created": 1441385679966,
        "componentType": "STANDARD",
        "ignoreQualifiers": 0,
        "importAutomatically": false,
        "useVfs": true,
        "active": true,
        "integrationFailed": false,
        "deleted": false,
        "defaultVersionType": "FULL",
        "cleanupDaysToKeep": 0,
        "cleanupCountToKeep": 0,
        "tags": [],
        "user": "admin"
      },
      "graphPosition": 3
    }
  ]
}
```


Related REST command: Get an application process request's activity trace..

getApplicationProcessRequest

Get information about a specific application process request

This command retrieves a specific application process request by id.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessRequest
```

Parameters

Table 911. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the application process request.

Related REST command: Get information about a specific application process request.

getApplicationProcessRequestProperties

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessRequestProperties
```

Parameters

Table 912. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the application process request.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProcessRequest
  -requestID 134t52le97-3f75-6783-333f-3b54d48c8d88
```

Example

Example response for a manual task approval


```
"tasks": [
     {
       "id": "19523fdc-3f75-6783-333f-3b54d48c8d88",
       "name": "Environment Approval for Test",
       "commentRequired": false,
       "commentPrompt": "Env MTask",
       "status": "OPEN",
       "startDate": 1740065588235,
       "userCanModify": true,
       "completedBy": "admin",
       "completedOn": 1740065601084,
       "comment": ""
     },
     {
       "id": "19523fdc-4284-db5d-add7-0a6af202bf1c",
       "name": "Application Approval for App",
       "commentRequired": false,
       "commentPrompt": "App MTask",
       "status": "OPEN",
       "startDate": 1740065588274,
       "userCanModify": true,
       "completedBy": "admin",
       "completedOn": 1740065595720,
       "comment": ""
     },
     {
       "id": "19523fdc-452a-c387-a66b-786513c18a88",
       "name": "Component Approval: Comp",
       "commentRequired": false,
       "commentPrompt": "Comp MTask",
       "status": "OPEN",
       "startDate": 1740065588306,
       "userCanModify": true,
       "completedBy": "admin",
       "completedOn": 1740065608814,
       "comment": ""
     }
   ]
```


Related REST command: Get application process request properties.

getApplicationProcessRequestStatus

Request the status of an application process request

Will return both the status of the request (which indicates if it is still running) and the request result (which indicates the result of the process if it completed).

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessRequestStatus [parameters]
```

Parameters

Table 913. Parameters

Parameter	Type	Required	Description
request	string	true	ID of the application process request

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProcessRequestStatus
  -request 47610d75-25c8-420d-a696-f006c34f5423
```

Example

Example response

```
{
  "status": "CLOSED",
  "result": "SUCCEEDED"
}
```

This command returns the following statuses:

- `CANCELING`
- `CLOSED`
- `COMPENSATING`
- `EXECUTING`
- `FAULTED`
- `FAULTING`
- `INITIALIZED`
- `PENDING`
- `UNINITIALIZED`

This command returns the following results:

- `APPROVAL REJECTED`
- `AWAITING APPROVAL`
- `CANCELED`
- `COMPENSATED`
- `FAILED TO START`
- `FAULTED`
- `NONE`

- `SCHEDULED FOR FUTURE`
- `SUCCEEDED`
- `UNINITIALIZED`

Related REST command: Request the status of an application process request.

getApplicationProcessRequestVersions

Get the Versions for an application process request

Get the Versions for an application process request

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessRequestVersions
```

Parameters

Table 914. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the application process request.

Related REST command: Get the Versions for an application process request.

getApplicationProcessUnfilledProperties

List properties that must be specified for a process

If a snapshot name or ID is specified, results include any component process steps which have some properties marked to receive values later for a specific version based on the snapshot.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProcessUnfilledProperties [parameters]
```

Parameters

Table 915. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Table 915. Parameters (continued)

Parameter	Type	Required	Description
processName	string	true	Name of the application process
snapshot	string	false	Name or ID of the snapshot (optional)
environment	string	false	Get the properties based on if they were deployed to the specified environment (optional)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProcessUnfilledProperties
  -application MyApplication
  -processName MyProcess
```

Example

Example response

```
[
  {
    "id": "decc3570-17ae-4b46-b640-6a7180edafa9",
    "name": "MyProperty",
    "label": "MyProperty",
    "pattern": "",
    "type": "TEXT",
    "value": "",
    "required": false,
    "description": "",
    "placeholder": ""
  }
]
```

Related REST command: List properties that must be specified for a process.

getApplicationProperties

Get all application properties and their values.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProperties [parameters]
```

Parameters

Table 916. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProperties
  -application JPetStore
```

Example

Example response

```
[
  {
    "id": "8b383d28-f64f-4f94-8ea1-1e85a323f527",
    "name": "Prop1",
    "value": "value1",
    "description": "",
    "secure": false
  },
  {
    "id": "e3a54008-6c73-4c72-b197-72b830b2ac6d",
    "name": "Prop2",
    "value": "value2",
    "description": "",
    "secure": false
  },
  {
    "id": "cce49826-a236-4bee-a8f4-cd015f2a6e8b",
    "name": "Prop3",
    "value": "value3",
    "description": "",
    "secure": false
  }
]
```

Related REST command: Get all application properties and their values..

getApplicationProperty

Get the value of a custom property on a application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationProperty [parameters]
```

Parameters

Table 917. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationProperty
  -application JPetStore
  -name Prop1
```

Related REST command: Get the value of a custom property on a application.

getApplicationTemplate

Get information about an application template

This command returns a JSON representation of an application template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationTemplate [parameters]
```

Parameters

Table 918. Parameters

Parameter	Type	Required	Description
applicationTemplate	string	true	Name or ID of the application template

Related REST command: Get information about an application template.

getApplicationTemplatePropDefs

Get information about the property definitions for the application template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationTemplatePropDefs [parameters]
```

Parameters

Table 919. Parameters

Parameter	Type	Required	Description
applicationTemplate	string	true	Name or ID of the application template

Related REST command: Get information about the property definitions for the application template.

getApplicationTemplates

Get information about all application templates on the server

This command returns a JSON representation of all application templates.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationTemplates
```

Parameters

None.

Related REST command: Get information about all application templates on the server.

getApplications

Get information about all applications on the server

This command returns a JSON representation of all applications.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplications
```

Parameters

None.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplications
```

Example

Example response

```
[
  {
    "id": "c8a45972-da39-4450-87f0-21fae710e0f5",
    "name": "JPetStore",
    "description": "",
    "created": 1387378155981,
    "enforceCompleteSnapshots": false,
    "active": true,
    "tags": [
    ],
    "user": "admin"
  },
  {
    "id": "b0d47a25-c90e-4286-8b9e-6a08a1da45bc",
    "name": "My Application",
    "description": "",
    "created": 1390502070483,
    "enforceCompleteSnapshots": false,
    "active": true,
    "tags": [
    ],
    "user": "admin"
  }
]
```

Related REST command: Get information about all applications on the server.

getApplicationsProcessRequests

Get information about all application process requests on the server

This command returns a JSON representation of all application process requests, up to a maximum of 100.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationsProcessRequests [parameters]
```

Parameters

Table 920. Parameters

Parameter	Type	Required	Description
startedAfter	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server includes the application process requests that were created after the specified timestamp.
startIndex	int	false	An index number that specifies the application process request to start with. You can use this index number to page through results.

Table 920. Parameters (continued)

Parameter	Type	Required	Description
useExecutionTime	boolean	false	Set to true to use execution start time instead of submission time for startedAfter

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationsProcessRequests
  -startedAfter 1471603999000
  startIndex 200
```

Example

Example response


```
[
  {
    "application": {
      "id": "962d3544-c028-4d44-8d0d-52311043e83d",
      "securityResourceId": "9543d94f-d9b9-4a34-a43a-f4d7d69b9987",
      "name": "MyApplication",
      "description": "",
      "created": 1471534973567,
      "enforceCompleteSnapshots": false,
      "active": true,
      "tags": [],
      "deleted": false,
      "user": "admin"
    },
    "applicationProcess": {
      "id": "4a38f91e-41bb-49c0-8010-d406d65fa7ba",
      "name": "deploy",
      "description": "",
      "active": true,
      "inventoryManagementType": "AUTOMATIC",
      "offlineAgentHandling": "PRE_EXECUTION_CHECK",
      "versionCount": 2,
      "version": 2,
      "commit": 17,
      "path":
 "applications/962d3544-c028-4d44-8d0d-52311043e83d/processes/4a38f91e-41bb-49c0-8010-d406d65fa7ba",
      "deleted": false,
      "metadataType": "applicationProcess"
    },
    "environment": {
      "id": "8ecf9ba2-4f7a-4326-a4bd-309a0366b2a2",
      "securityResourceId": "fdab7788-519a-4118-87a7-d1e4f0f391c9",
```

```
      "name": "local",
      "description": "",
      "color": "#00B2EF",
      "requireApprovals": false,
      "noSelfApprovals": false,
      "lockSnapshots": false,
      "calendarId": "27c132c9-b805-4ece-b6f9-265a1b355c21",
      "active": true,
      "deleted": false,
      "cleanupDaysToKeep": 0,
      "cleanupCountToKeep": 0,
      "enableProcessHistoryCleanup": false,
      "useSystemDefaultDays": true,
      "historyCleanupDaysToKeep": 365,
      "conditions": []
    },
    "id": "3bc33fb8-98a6-4a24-a97b-d6e4b4b541a3",
    "submittedTime": 1471535097984,
    "traceId": "46749e41-d571-4b3c-a3b2-6fd0b6f23858",
    "userName": "admin",
    "onlyChanged": true,
    "description": "",
    "startTime": 1471535098738,
    "result": "SUCCEEDED",
    "state": "CLOSED",
    "paused": false,
    "endTime": 1471535099134,
    "duration": 396
  },
  {
    "application": {
      "id": "8ba25265-f313-4246-aa8a-4f8a9617ed2b",
      "securityResourceId": "10935be7-678d-4dc6-bbd1-57644d619041",
      "name": "MyApplication2",
      "description": "",
      "created": 1471535176734,
      "enforceCompleteSnapshots": false,
      "active": true,
      "tags": [],
      "deleted": false,
      "user": "admin"
    },
    "applicationProcess": {
      "id": "4d4d727c-a92e-45f0-8901-1277e92c09f4",
      "name": "deploy",
      "description": "",
      "active": true,
      "inventoryManagementType": "AUTOMATIC",
      "offlineAgentHandling": "PRE_EXECUTION_CHECK",
      "versionCount": 2,
      "version": 2,
      "commit": 20,
      "path":
"applications/8ba25265-f313-4246-aa8a-4f8a9617ed2b/processes/4d4d727c-a92e-45f0-8901-1277e92c09f4",
      "deleted": false,
      "metadataType": "applicationProcess"
    },
    "environment": {
```

```
      "id": "03eae7bf-f97f-4e2d-81b5-8780ee1d5277",
      "securityResourceId": "43fa87b2-1751-44d8-9abe-e788d7990608",
      "name": "local",
      "description": "",
      "color": "#00B2EF",
      "requireApprovals": false,
      "noSelfApprovals": false,
      "lockSnapshots": false,
      "calendarId": "79fc6057-f0a6-42a1-be0f-939ac76dbd50",
      "active": true,
      "deleted": false,
      "cleanupDaysToKeep": 0,
      "cleanupCountToKeep": 0,
      "enableProcessHistoryCleanup": false,
      "useSystemDefaultDays": true,
      "historyCleanupDaysToKeep": 365,
      "conditions": []
    },
    "id": "0596ac0b-36c0-449a-a11b-a5aa0cccb98b",
    "submittedTime": 1471535228001,
    "traceId": "e10524ee-e43c-44ea-9a5b-610d6dcbf2e8",
    "userName": "admin",
    "onlyChanged": true,
    "description": "",
    "startTime": 1471535228400,
    "result": "SUCCEEDED",
    "state": "CLOSED",
    "paused": false,
    "endTime": 1471535228502,
    "duration": 102
  }
]
```


Related REST command: Get information about all application process requests on the server.

getApplicationsProcessRequestsCount

Count all application process requests

This command returns the total number of application process requests for all applications.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApplicationsProcessRequestsCount [parameters]
```

Parameters

Table 921. Parameters

Parameter	Type	Required	Description
startedAfter	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server counts all application process requests that were created after the specified timestamp.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getApplicationsProcessRequestsCount
  -startedAfter 1471603999000
```

Example

Example response

```
{
  "appProcReqCount": 223
}
```

Related REST command: Count all application process requests.

getApprovalTasks

List recent approval tasks

This command returns the approval tasks that have been modified since the given time.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getApprovalTasks [parameters]
```

Parameters

Table 922. Parameters

Parameter	Type	Required	Description
modifiedAfter	long	false	Include approval tasks that have been created or edited since this provided time. (Optional)

Related REST command: List recent approval tasks.

getAuthToken

Get details for an authentication token

Returns the details for a specific authentication token identified by database ID or token secret. Only metadata for the token is returned; the token secret itself is not available. Only users with 'Manage Security' permission can use this function for a token they do not own.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthToken [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"token": "Authentication token ID or secret"}
```

Example

Example command to get auth tokens using a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAuthToken filename.json
```

Example

JSON example

```
{
  "token": "194dba21-0199-cddb-1f04-4ad922fc5385"
}
```

Related REST command: Get details for an authentication token.

getAuthTokens

Get all auth tokens

Returns a page of authentication token details out of the list of all tokens. Only metadata for the tokens is returned; the token secrets themselves are not available. Pages are numbered starting at 1 and a page can range from 1 to 100 tokens. Only users with 'Manage Security' permission can use this function.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthTokens [parameters]
```

Parameters

Table 923. Parameters

Parameter	Type	Required	Description
page	int	true	Page number starting at 1
pageSize	int	true	Page size starting at 1 and to at most 100

Example

Example command to get auth tokens by pagination

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAuthTokens
  -page 1
  -pageSize 10
```

Related REST command: Get all auth tokens.

getAuthTokensForUser

Get active authentication tokens for user

Returns the list of authorization tokens for a user. Only metadata for the tokens is returned; the token secrets themselves are not available. Only users with 'Manage Security' permission can use this function for users other than themselves.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthTokensForUser [parameters]
```

Parameters

Table 924. Parameters

Parameter	Type	Required	Description
user	string	false	The user for which tokens are returned. Required if querying tokens for a user other than the one making the request.

Example

Example command to get auth tokens of a specific user

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getAuthTokensForUser
  -user jsmith
```

Example

Example response for a specific user

```
{
  "tokens": [{
    "id": "194dba1b-ff4a-473e-19a8-c96d1246b690",
    "expiration": 1740208500000,
    "host": "",
    "description": "dummyuser",
    "createdDate": 1738851598324,
    "verbToUrls": [{
      "verb": "ALL",
      "url": "*"
    }]
  }],
  "userId": "a51fe9df-9625-48c2-bd01-4b50acf4c72b",
  "userName": "jsmith"
}
```

Related REST command: Get active authentication tokens for user.

getAuthenticationRealm

Get information about an AuthenticationRealm

This command returns a JSON representation of an AuthenticationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthenticationRealm
```

Parameters

Table 925. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the AuthenticationRealm

Related REST command: Get information about an AuthenticationRealm.

getAuthenticationRealms

Get information about all AuthenticationRealms

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthenticationRealms
```

Parameters

None.

Related REST command: Get information about all AuthenticationRealms.

getAuthorizationRealm

Get information about an AuthorizationRealm

This command returns a JSON representation of an AuthorizationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthorizationRealm [parameters]
```

Parameters

Table 926. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the AuthorizationRealm

Related REST command: Get information about an AuthorizationRealm.

getAuthorizationRealms

Get information about all AuthorizationRealms

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAuthorizationRealms
```

Parameters

None.

Related REST command: Get information about all AuthorizationRealms.

getAvailablePlugins

Get a list of all available plugins for installing into the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getAvailablePlugins
```

Parameters

None.

Related REST command: Get available plug-ins.

getBlueprintNodePropertiesTemplate

Get blueprint properties

This command returns a JSON template of the properties that are required to provision a blueprint.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getBlueprintNodePropertiesTemplate [parameters]
```

Parameters

Table 927. Parameters

Parameter	Type	Required	Description
blueprint	string	true	Name or ID of the blueprint
application	string	false	Name of the application; this parameter is required if you specify the blueprint name instead of ID

Related REST command: Get blueprint properties.

getBlueprintsInApplication

Get all blueprints in an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getBlueprintsInApplication [parameters]
```

Parameters

Table 928. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getBlueprintsInApplication
  -application JPetStore
```

Example

Example response

```
[
  {
    "id": "7e0a196f-41ca-40e8-aed8-4f47cafcedc9",
    "name": "My blueprint",
    "description": ""
  },
  {
    "id": "362cfc88-e5f7-4d98-aaee-a53edec5c8d2",
    "name": "Other blueprint",
    "description": ""
  }
]
```

Related REST command: Get all blueprints in an application.

getComponent

Get information about a component

This command returns a JSON representation of a component.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponent [parameters]
```

Parameters

Table 929. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponent
  -component JPetStore-APP
```

Example

Example response


```
{
  "id": "69a6a5bd-583b-4b40-bd47-54b28f5aa1e6",
  "name": "JPetStore-APP",
  "description": "",
  "created": 1387315824908,
  "importAutomatically": false,
  "useVfs": true,
  "active": true,
  "deleted": false,
  "defaultVersionType": "FULL",
  "cleanupDaysToKeep": 0,
  "cleanupCountToKeep": 0,
  "tags": [
    {
      "id": "d140ae5a-dfb0-477a-910b-762cc0d99cc6",
      "name": "Tag2",
      "color": "#7fff00",
      "description": "",
      "objectType": "Component"
    }
  ],
  "user": "admin",
  "properties": [
    {
      "id": "91b61a4a-1454-46d9-af32-eb98e6ae7d3d",
      "name": "code_station/repository",
      "value": "9f19c4c5-84b9-4f4e-b1fc-df2c2f306b1c",
      "description": "",
      "secure": false
    },
    {
      "id": "3702f391-09e1-4550-903a-d5ae286c94aa",
      "name": "custom/Prop1",
      "value": "value1",
      "description": "",
      "secure": false
    },
    {
      "id": "4fcb6120-453d-45a5-8cb3-9e7945dc80a5",
      "name": "FileSystemVersionedComponentProperties/basePath",
      "value": "/root/shared/app",
      "description": "",
      "secure": false
    },
    {
      "id": "8f3706d1-8255-4626-b349-6e503a1ed4e4",
      "name": "FileSystemVersionedComponentProperties/extensions",
      "value": "",
      "description": "",
      "secure": false
    },
    {
      "id": "8bba5e9f-7937-4b24-ac12-ebd6250f0023",
      "name": "FileSystemVersionedComponentProperties/saveFileExecuteBits",
```

```
      "value": "false",
      "description": "",
      "secure": false
    }
  ],
  "applications": [
    {
      "id": "c8a45972-da39-4450-87f0-21fae710e0f5",
      "name": "JPetStore",
      "description": "",
      "created": 1387378155981,
      "enforceCompleteSnapshots": false,
      "active": true,
      "tags": [
      ],
      "user": "admin"
    }
  ],
  "environmentPropSheetDef": {
    "id": "8f9bf513-d434-43d4-ada4-6b8d16cfe4c2",
    "path": "components/69a6a5bd-583b-4b40-bd47-54b28f5aa1e6/environmentPropSheetDef",
    "version": 1,
    "versionCount": 1,
    "commit": 6,
    "versioned": true
  },
  "versionPropSheetDef": {
    "id": "99bb8cce-44ee-49df-bebc-e84dbbdba2ac",
    "path": "components/69a6a5bd-583b-4b40-bd47-54b28f5aa1e6/versionPropSheetDef",
    "version": 1,
    "versionCount": 1,
    "commit": 6,
    "versioned": true
  },
  "propSheet": {
    "id": "cb6049f9-bcce-4806-9d24-de5d729f40b3",
    "name": "custom",
    "path":
 "components/69a6a5bd-583b-4b40-bd47-54b28f5aa1e6/propSheetGroup/propSheets/cb6049f9-bcce-4806-9d24-de5d729f40b
3",
    "version": 2,
    "versionCount": 2,
    "commit": 87,
    "versioned": true
  },
  "templatePropSheet": {
    "id": "e94a21c9-138c-4c36-91a0-ddc9d6f5cf07",
    "name": "template",
    "path":
 "components/69a6a5bd-583b-4b40-bd47-54b28f5aa1e6/propSheetGroup/propSheets/e94a21c9-138c-4c36-91a0-ddc9d6f5cf0
7",
    "version": 1,
    "versionCount": 1,
    "commit": 6,
    "versioned": true
  },
  "resourceRole": {
    "id": "06c00bb6-5ff0-4960-867f-5fb8d2e69222",
```

```
    "name": "JPetStore-APP",
    "specialType": "COMPONENT",
    "propSheetDef": {
      "id": "100de7fc-182f-4fcf-8299-46fae5a06935",
      "name": "JPetStore-APP",
      "versioned": false
    },
    "propDefs": [
      {
        "id": "bb8404e1-c030-436e-ba2f-c2616216cf18",
        "name": "myProp",
        "label": "myProp",
        "pattern": "",
        "type": "TEXT",
        "value": "42",
        "required": false,
        "description": ""
      }
    ]
  },
  "systemCleanupDaysToKeep": -1,
  "systemCleanupCountToKeep": -1,
  "extendedSecurity": {
    "read": true,
    "write": true,
    "Create Components": true,
    "Edit Components": true,
    "Manage Versions": true,
    "View Components": true,
    "teams": [
    ]
  }
```


Related REST command: Get information about a component.

getComponentEnvironmentProperties

Get component environment properties for an environment

These properties are defined on the component and their values are set on the environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentEnvironmentProperties [parameters]
```

Parameters

Table 930. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment

Table 930. Parameters (continued)

Parameter	Type	Required	Description
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
component	string	true	Name or ID of the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentEnvironmentProperties
  -component JPetStore-APP
  -environment 1f97af86-cde3-4ba2-803b-0e4580dcc05b
```

Example

Example response

```
[
  {
    "id": "c598738e-2057-44d7-aee7-e805df40cc69",
    "name": "Environment property 1",
    "value": "false",
    "description": "",
    "secure": false
  }
]
```

Related REST command: Get component environment properties for an environment.

getComponentProcess

Get information about a component process

This command returns a JSON representation of a component process.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProcess [parameters]
```

Parameters

Table 931. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
componentProcess	string	true	Name or ID of the component process
draft	boolean	false	Whether to get the latest draft process. Default is false

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentProcess
  -component JPetStore-APP
  -componentProcess "Deploy application component"
```

Example

Example response

```
{
  "id": "c59721d1-a02c-4b3f-9eb7-bde5358210d0",
  "name": "Deploy application component",
  "description": "",
  "defaultWorkingDir": "${p:resource/work.dir}/${p:component.name}",
  "takesVersion": true,
  "inventoryActionType": "ADD",
  "status": "Active",
  "configActionType": "ADD",
  "active": true,
  "versionCount": 4,
  "version": 4,
  "commit": 32,
  "path": "components/69a6a5bd-583b-4b40-bd47-54b28f5aa1e6/processes/c59721d1-a02c-4b3f-9eb7-bde5358210d0"
}
```

Related REST command: Get information about a component process.

getComponentProcessRequest

Get information about a specific component process request

Generally used as an endpoint to retrieve APRs that were scheduled in advance.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProcessRequest
```

Parameters

Table 932. Parameters

Parameter	Type	Required	Description
requestID	string	true	The ID of the component process request.

Related REST command: Get information about a specific component process request.

getComponentProcessRequestCount

Get the number of component process requests submitted after a certain time given a specified component

This command returns a count of all the component process requests of a specific componentsubmitted after a certain time.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProcessRequestCount [parameters]
```

Parameters

Table 933. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
submittedAfter	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server counts all CPRs that were created after the specified timestamp.

Related REST command: Get the number of component process requests submitted after a certain time given a specified component.

getComponentProcessRequests

Get information about all component process requests

This command returns a JSON representation of a list of up to 100 component process requests at a time.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProcessRequests [parameters]
```

Parameters

Table 934. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
submittedAfter	long	false	Time value that marks the oldest CPR we want to retrieve.

Related REST command: Get information about all component process requests.

getComponentProperties

Get values for component properties

This command returns all component properties.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProperties [parameters]
```

Parameters

Table 935. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentProperties
  -component JPetStore-APP
```

Example

Example response

```
[
  {
```

```
    "id": "3702f391-09e1-4550-903a-d5ae286c94aa",
    "name": "Prop1",
    "value": "value1",
    "description": "",
    "secure": false
  }
]
```

Related REST command: Get values for component properties.

getComponentProperty

Get the value of a property on a component. Property names can be prefixed to specify which set of propertieswill be searched. Unprefixed properties will search custom 'Component Properties' on a component while a propertynamed such as template/propertyName will search the properties inherited from a component templates property definitions.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentProperty [parameters]
```

Parameters

Table 936. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the property.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentProperty
  -component JPetStore-APP
  -name Prop1
```

Related REST command: Get the value of a property on a component.

getComponentTemplatePropDefs

Get information about the property definitions for the component template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentTemplatePropDefs [parameters]
```

Parameters

Table 937. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template

Related REST command: Get information about the property definitions for the component template.

getComponentTemplates

Get information about all component templates on the server

This command returns a JSON representation of all component templates.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentTemplates
```

Parameters

None.

Related REST command: Get information about all component templates on the server.

getComponentVersionPropDefs

List the version property definitions for a component

These properties are available to each version of the component.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentVersionPropDefs [parameters]
```

Parameters

Table 938. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentVersionPropDefs
  -component JPetStore-APP
```

Example

Example response

```
[
  {
    "id": "870bc2f5-55e2-4cb9-b3ab-f750688754b9",
    "name": "Prop1",
    "label": "Prop1",
    "pattern": "",
    "type": "CHECKBOX",
    "value": "true",
    "required": false,
    "description": ""
  },
  {
    "id": "ee02cbf0-9fc9-465e-a37f-754c40222d0d",
    "name": "Prop2",
    "label": "",
    "pattern": "",
    "type": "TEXT",
    "value": "prop2",
    "required": false,
    "description": ""
  }
]
```

Related REST command: List the version property definitions for a component.

getComponentVersions

Get the versions of a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentVersions [parameters]
```

Parameters

Table 939. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Table 939. Parameters (continued)

Parameter	Type	Required	Description
inactive	boolean	false	Specify true to include archived versions; the default is false
numResults	int	false	Specify a number greater than 0 to limit the number of results. If numResults is not specified, or is given to be < 1, no limit is applied.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentVersions
  -component MyComponent
```

Example

Example response

```
[
  {
    "id": "1a53a4a6-bfcd-4014-a0ab-83f97077a4d4",
    "name": "1",
    "description": "",
    "type": "FULL",
    "created": 1441381421819,
    "active": true,
    "archived": false,
    "sizeOnDisk": 4096
  },
  {
    "id": "ed2361a4-948c-4646-bda4-7465df0e38b5",
    "name": "2",
    "description": "",
    "type": "FULL",
    "created": 1441381420863,
    "active": true,
    "archived": false,
    "sizeOnDisk": 4096
  }
]
```

Related REST command: Get the versions of a component.

getComponents

Get information about all components on the server

This command returns a JSON representation of all components on the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponents [parameters]
```

Parameters

Table 940. Parameters

Parameter	Type	Required	Description
active	boolean	false	

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponents
```

Example

Example response

```
[
  {
    "id": "69a6a5bd-583b-4b40-bd47-54b28f5aa1e6",
    "name": "JPetStore-APP",
    "description": "",
    "created": 1387315824908,
    "importAutomatically": false,
    "useVfs": true,
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
      {
        "id": "d140ae5a-dfb0-477a-910b-762cc0d99cc6",
        "name": "Tag2",
        "color": "#7fff00",
        "description": "",
        "objectType": "Component"
      }
    ],
    "user": "admin"
  },
  {
    "id": "f4a37f52-7d96-4fb5-b43b-57166b177e7c",
    "name": "JPetStore-DB",
    "description": "",
    "created": 1387316389625,
    "importAutomatically": false,
    "useVfs": true,
```

```
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
    ],
    "user": "admin"
  },
  {
    "id": "76299d8e-c328-42f4-99ab-aa7bfee7aa14",
    "name": "JPetStore-WEB",
    "description": "",
    "created": 1387316469977,
    "importAutomatically": false,
    "useVfs": true,
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
    ],
    "user": "admin"
  }
]
```


Related REST command: Get information about all components on the server.

getComponentsInApplication

Get all components in an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getComponentsInApplication [parameters]
```

Parameters

Table 941. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getComponentsInApplication
  -application JPetStore
```

Example

Example response


```
[
  {
    "id": "69a6a5bd-583b-4b40-bd47-54b28f5aa1e6",
    "name": "JPetStore-APP",
    "description": "",
    "created": 1387315824908,
    "importAutomatically": false,
    "useVfs": true,
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
    ],
    "user": "admin"
  },
  {
    "id": "f4a37f52-7d96-4fb5-b43b-57166b177e7c",
    "name": "JPetStore-DB",
    "description": "",
    "created": 1387316389625,
    "importAutomatically": false,
    "useVfs": true,
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
    ],
    "user": "admin"
  },
  {
    "id": "76299d8e-c328-42f4-99ab-aa7bfee7aa14",
    "name": "JPetStore-WEB",
    "description": "",
    "created": 1387316469977,
    "importAutomatically": false,
    "useVfs": true,
    "active": true,
    "deleted": false,
    "defaultVersionType": "FULL",
    "cleanupDaysToKeep": 0,
    "cleanupCountToKeep": 0,
    "tags": [
    ],
    "user": "admin"
  }
]
```


Related REST command: Get all components in an application.

getConfigurableAgentLicenseTypes

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getConfigurableAgentLicenseTypes
```

Parameters

None.

Related REST command: Get configurable agent license types.

getCurrentActivityList

Returns all child workflows of a parent request id if specified else returns all running workflow activities.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getCurrentActivityList [parameters]
```

Parameters

Table 942. Parameters

Parameter	Type	Required	Description
parentRequestId	string	false	Parent request id. If specified, this returns any child workflows running under the process request.

Related REST command: Get all child workflows of a parent request id.

getDatabaseLatency

Gather statistics by runnning a database latency test. Result times are in milliseconds. This may take several seconds.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getDatabaseLatency
```

Parameters

None.

Related REST command: Gather statistics by runnning a database latency test.

getDeploymentTrigger

Get information about a single deployment trigger.

This command returns a JSON representation of a single deployment trigger.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getDeploymentTrigger
```

Parameters

Table 943. Parameters

Parameter	Type	Required	Description
deploymentTriggerId	string	true	ID of the deployment trigger.

Related REST command: Get information about a single deployment trigger..

getDeploymentTriggers

Get information about all deployment triggers for an environment.

This command returns a JSON representation of all deployment triggers for an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getDeploymentTriggers [parameters]
```

Parameters

Table 944. Parameters

Parameter	Type	Required	Description
environment	string	true	ID of the environment.

Related REST command: Get information about all deployment triggers for an environment..

getEnvironment

Get information about an environment

This command returns a JSON representation of an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironment [parameters]
```

Parameters

Table 945. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name of the application; this value is required if you specify the environment name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getEnvironment
  -application JPetStore
  -environment "Tutorial environment 1"
```

Example

Example response


```
{
  "id": "1f97af86-cde3-4ba2-803b-0e4580dcc05b",
  "name": "Tutorial environment 1",
  "description": "",
  "color": "#ffffff",
  "requireApprovals": false,
  "lockSnapshots": false,
  "calendarId": "e237199e-8bbc-497e-97eb-ca2568dff91c",
  "active": true,
  "cleanupDaysToKeep": 0,
  "cleanupCountToKeep": 0,
  "conditions": [
  ],
  "compliancy": {
    "missingCount": 0,
    "correctCount": 3,
    "desiredCount": 3
  },
  "systemCleanupDaysToKeep": -1,
  "systemCleanupCountToKeep": -1,
  "exemptProcesses": "",
  "extendedSecurity": {
    "read": true,
    "write": true,
    "execute": true,
    "Create Environments": true,
    "Edit Environments": true,
    "Execute on Environments": true,
    "View Environments": true,
```

```
    "teams": [
      {
        "teamId": "ea307e2e-d6be-4092-a7e8-e8e02274de7e",
        "teamLabel": "My team"
      }
    ]
  }
}
```

Related REST command: Get information about an environment.

getEnvironmentBaseResources

Get all base resources for an environment

This command returns a listing of all resources added as base resources to an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentBaseResources [parameters]
```

Parameters

Table 946. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name of the application; this value is required if you specify the environment name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getEnvironmentBaseResources
  -application JPetStore
  -environment "Tutorial environment 1"
```

Example

Example response

```
[
  {
    "id": "4db385d8-34bb-4625-8e62-c627175a2cf7",
    "name": "Tutorial agent 1",
    "path": "/JPetStore agents/Tutorial agent 1",
    "active": true,
```

```
    "description": "",
    "inheritTeam": true,
    "impersonationPassword": "****",
    "impersonationUseSudo": false,
    "impersonationForce": false,
    "type": "agent",
    "status": "ONLINE",
    "hasAgent": true,
    "tags": [
    ]
  }
]
```

Related REST command: Get all base resources for an environment.

getEnvironmentCompliancy

Get an inventory compliancy report for an environment.

Returns an inventory compliancy report for an environment. By default, it returns the complete inventory organized by component and then by version. Alternatively, it can return a summary of inventory compliancy counts

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentCompliancy [parameters]
```

Parameters

Table 947. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; optional if the environment is referenced by ID
summary	boolean	false	True to get a summary of environment compliance. The default is false.

Related REST command: Get an inventory compliancy report for an environment..

getEnvironmentGate

Get the environment gate of a given environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentGate [parameters]
```

Parameters

Table 948. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of environment
application	string	false	Name or ID of application (required if environment is specified by name)

Related REST command: Get the environment gate of a given environment.

getEnvironmentProperties

Get environment property values

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentProperties [parameters]
```

Parameters

Table 949. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getEnvironmentProperties
  -application JPetStore
  -environment "Tutorial environment 1"
```

Example

Example response

```
[
  {
```

```
    "id": "9dbcf2f6-a96e-4ee5-ba65-07c5fa8422c9",
    "name": "db.url",
    "value": "jdbc:mysql://localhost:3306/jpetstore",
    "description": "",
    "secure": false
  },
  {
    "id": "f395d093-95d8-4d1d-8901-fc0329ef6bdb",
    "name": "tomcat.home",
    "value": "/opt/apache-tomcat-6.0.37",
    "description": "",
    "secure": false
  },
  {
    "id": "404fb6e0-b0a9-433a-9b63-d8614017d459",
    "name": "tomcat.manager.url",
    "value": "http://localhost:8080/manager",
    "description": "",
    "secure": false
  },
  {
    "id": "167fa275-bb95-460b-97b2-e27e713a1400",
    "name": "tomcat.start",
    "value": "/opt/apache-tomcat-6.0.37/bin/startup.sh",
    "description": "",
    "secure": false
  }
]
```


Related REST command: Get environment property values.

getEnvironmentProperty

Get the value of a custom property on an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentProperty [parameters]
```

Parameters

Table 950. Parameters

Parameter	Type	Required	Description
name	string	true	Name of the property
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getEnvironmentProperty
  -application JPetStore
  -environment "Tutorial environment 1"
  -name tomcat.start
```

Related REST command: Get the value of a custom property on an environment..

getEnvironmentResourceTree

Get resource tree for an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentResourceTree [parameters]
```

Parameters

Table 951. Parameters

Parameter	Type	Required	Description
includeSecurity	boolean	false	Include security json. Not including security json can be much faster. Defaults to true.
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application. This is required when using environment name.

Related REST command: Get resource tree for an environment.

getEnvironmentTemplate

Get information about an environment template

This command returns a JSON representation of an environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentTemplate [parameters]
```

Parameters

Table 952. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	ID of the environment template

Related REST command: Get information about an environment template.

getEnvironmentTemplateGate

Get the environment gate of a given environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentTemplateGate [parameters]
```

Parameters

Table 953. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	Name or ID of environment template
applicationTemplate	string	false	Name or ID of applicationTemplate (required if environment template is specified by name)

Related REST command: Get the environment gate of a given environment template.

getEnvironmentTemplatePropDefs

Get information about the property definitions for the environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentTemplatePropDefs [parameters]
```

Parameters

Table 954. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	ID of the environment template

Related REST command: Get information about the property definitions for the environment template.

getEnvironmentsInApplication

Get information about all environments in an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getEnvironmentsInApplication [parameters]
```

Parameters

Table 955. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getEnvironmentsInApplication
  -application JPetStore
```

Example

Example response

```
[
  {
    "id": "1f97af86-cde3-4ba2-803b-0e4580dcc05b",
    "name": "Tutorial environment 1",
    "description": "",
    "color": "#ffffff",
    "requireApprovals": false,
    "lockSnapshots": false,
    "calendarId": "e237199e-8bbc-497e-97eb-ca2568dff91c",
    "active": true,
    "cleanupDaysToKeep": 0,
```

```
    "cleanupCountToKeep": 0,
    "conditions": [
    ]
  }
]
```

Related REST command: Get information about all environments in an application.

getExternalApprovalProcess

Get information about an external approval process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getExternalApprovalProcess [parameters]
```

Parameters

Table 956. Parameters

Parameter	Type	Required	Description
externalApprovalProcess	string	true	Name or ID of the external approval process

Related REST command: Get information about an external approval process.

getExternalApprovalProcesses

Get all external approval processes

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getExternalApprovalProcesses
```

Parameters

None.

Related REST command: Get all external approval processes.

getFirstUserWithEmail

Get the first user with a given e-mail address

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getFirstUserWithEmail [parameters]
```

Parameters

Table 957. Parameters

Parameter	Type	Required	Description
email	string	true	E-mail address of the user

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  firstUserWithEmail
  -email jsmith@us.ibm.com
```

Example

Example response

```
{"id":"2c1c9012-2111-4d6a-b5f0-b6cabff94713","name":"janesmith"}
```

Related REST command: Get the first user with a given e-mail address.

getFrontendServers

Get a list of frontend servers.

Get information about all frontend servers.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getFrontendServers
```

Parameters

None.

Related REST command: Get a list of frontend servers..

getHighWatermarks

Return high watermarks for agent usage

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getHighWatermarks
```

Parameters

None.

Related REST command: Return high watermarks for agent usage.

getInstalledPlugins

Get a list of all currently installed plugins.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getInstalledPlugins
```

Parameters

None.

Related REST command: Get installed plug-ins.

getInventoryChangesForSnapshot

Get inventory changes for application process run with snapshot in environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getInventoryChangesForSnapshot [parameters]
```

Parameters

Table 958. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of snapshot
application	string	true	Name or ID of the application
environment	string	true	Name or ID of environment
applicationProcess	string	true	Name or ID of application process

Related REST command: Get inventory changes for application process run with snapshot in environment.

getJavaPackage

Get available Java packages.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getJavaPackage [parameters]
```

Parameters

Table 959. Parameters

Parameter	Type	Required	Description
package	string	true	Name or ID of the java package

Related REST command: Get available Java packages..

getJavaPackages

Get available Java packages.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getJavaPackages
```

Parameters

None.

Related REST command: Get available Java packages..

getJavaPackagesForAgent

Get Java packages compatible with a specific agent.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getJavaPackagesForAgent [parameters]
```

Parameters

Table 960. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Related REST command: Get Java packages compatible with a specific agent..

getLLMVendorModels

Gets the list of supported models for the given vendor.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getLLMVendorModels [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "accessTokenEndpoint": "The uri for getting an access
  token for authenticating with the LLM if neccessary.",
  "apiKey": "The API Key provided by the LLM Vendor. This
  may be required for Vendors such as Watsonx and OpenAI.",
  "endpoint": "The base URI pointing to the LLM provider
  API. Required.",
  "organization": "The organization if exists.",
  "project": "The project id if exists.",
  "vendor": "The name of the LLM Vendor. Options for this
  are returned by the getVendors endpoint. Required."
}
```

Related REST command: Gets the list of supported models for the given vendor..

getLLMVendors

Gets the list of supported LLM vendors.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getLLMVendors
```

Parameters

None.

Related REST command: Gets the list of supported LLM vendors..

getLatestDesiredInventory

Get the latest desired inventory for an environment.

This command returns a JSON array of all of the desired inventory for an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getLatestDesiredInventory
```

Parameters

Table 961. Parameters

Parameter	Type	Required	Description
environmentId	string	true	Name or ID of the environment

Related REST command: Get the latest desired inventory for an environment..

getLatestSnapshotForEnvironment

Get the latest snapshot deployed to an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getLatestSnapshotForEnvironment [parameters]
```

Parameters

Table 962. Parameters

Parameter	Type	Required	Description
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
environment	string	true	Name or ID of the environment

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getLatestSnapshotForEnvironment
  -application MyApplication
  -environment MyEnvironment
```

Example

Example response

```
{
  "id": "2ef3189b-af8f-49f1-8476-a3d4ed440722",
  "name": "Snap5",
  "created": 1441388592826,
  "active": true,
  "versionsLocked": false,
  "configLocked": false,
  "applicationId": "2d741420-fd31-443d-95eb-472d334f8ec9",
  "user": "admin"
}
```

Related REST command: Get the latest snapshot deployed to an environment.

getLoggingConfig

Get logging configuration

Returns the current logging configuration. The output is suitable for input to the setLoggingConfig command.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getLoggingConfig
```

Parameters

None.

Related REST command: Get logging configuration.

getMaintenanceBlackouts

Returns a list of all real and synthetic maintenance blackouts that start or end within the specified range. If no range is provided then all real maintenance blackouts will be returned (synthetic blackouts are not returned as the result would be infinite).

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getMaintenanceBlackouts [parameters]
```

Parameters

Table 963. Parameters

Parameter	Type	Required	Description
rangeStart	string	false	Start time of the search range. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time. (Optional) If rangeStart is provided then rangeEnd must also be provided

Table 963. Parameters (continued)

Parameter	Type	Required	Description
rangeEnd	string	false	End time of the search range. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time. (Optional)

Related REST command: Get a list of all real and synthetic maintenance blackouts that start or end within a specified range.

getOpenIDConnectRealms

Get all realms that support OpenID Connect login.

This command returns a list of all enabled authentication realms that support OpenID Connect login. This includes LDAP realms with the OpenID login mechanism configured.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getOpenIDConnectRealms
```

Parameters

None.

Related REST command: Get all realms that support OpenID Connect login..

getOverlappingArtifacts

Get all artifacts overlapped in the given version

Get all artifacts that are overlapped by later versions. A version is considered safe to rollback if it does not contain overlapping artifacts. It is risky to roll back a version if that version contains overlapping artifacts. This command applies only to z/OS component versions.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getOverlappingArtifacts
```

Parameters

Table 964. Parameters

Parameter	Type	Required	Description
resourceId	string	true	Name or ID of the resource

Table 964. Parameters (continued)

Parameter	Type	Required	Description
versionId	string	true	Name or ID of the version to check risky status
environmentId	string	true	Name or ID of the environment
componentId	string	true	Name or ID of the component

Related REST command: Get all artifacts overlapped in the given version.

getPostProcessingScript

Get information about a post processing script

This command returns a JSON representation of a post processing script.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getPostProcessingScript [parameters]
```

Parameters

Table 965. Parameters

Parameter	Type	Required	Description
script	string	true	Name/ID of the script

Related REST command: Get information about a post processing script.

getPostProcessingScripts

Get a list of post processing scripts.

Get information about post processing scripts.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getPostProcessingScripts
```

Parameters

None.

Related REST command: Get a list of post processing scripts..

getPreferredJavaPackageForAgent

Get the preferred Java package compatible with an agent.

This command gets the preferred Java package compatible with the platform of an agent. If no such package is currently defined, then no content is returned.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getPreferredJavaPackageForAgent [parameters]
```

Parameters

Table 966. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Related REST command: Get the preferred Java package compatible with an agent..

getProcessRequest

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getProcessRequest
```

Parameters

Table 967. Parameters

Parameter	Type	Required	Description
requestId	string	true	Id of process request

Related REST command: Get process request from request ID.

getProcessRequests

Get information about all generic process requests on the server

This command returns a JSON representation of all generic process requests, up to a maximum of 100.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getProcessRequests [parameters]
```

Parameters

Table 968. Parameters

Parameter	Type	Required	Description
startTime	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server includes the process requests that were created after the specified timestamp. Defaults to 24 hours ago.
endTime	long	false	A timestamp, in milliseconds from the UNIX epoch. If you specify this parameter, the server includes the process requests that were created before the specified timestamp. Defaults to the current time.
startIndex	int	false	An index number that specifies the application process request to start with. You can use this index number to page through results.

Related REST command: Get information about all generic process requests on the server.

getRelayStatuses

Returns a JSONObject with count of agent relays to status. Expected format: { "ONLINE":4, "OFFLINE":1, } Requires Settings Tab permission.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getRelayStatuses
```

Parameters

None.

Related REST command: Returns a JSONObject with count of agent relays to status.

getRelays

Get information about all relays a user has access to.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getRelays
```

Parameters

None.

Related REST command: Get information about all relays a user has access to..

getResource

Get information about a resource

This command returns a JSON representation of a resource, including its security settings.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResource [parameters]
```

Parameters

Table 969. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResource
  -resource "/My resource"
```

Example

Example response

```
{
  "id": "e7a3eb3c-a0b2-49d3-80ae-ce788edb101f",
  "name": "My resource",
  "path": "/My resource",
  "active": true,
  "description": "New resource for command example(String)",
  "inheritTeam": false,
  "impersonationPassword": "****",
  "impersonationUseSudo": false,
  "impersonationForce": false,
  "hasAgent": false,
  "tags": [
  ],
  "security": {
    "read": true,
    "write": true
```

```
  },
  "propSheet": {
    "id": "b00c0b1c-d007-4c19-a9dd-2e1397df8f1f",
    "name": "custom",
    "path":
 "resources/e7a3eb3c-a0b2-49d3-80ae-ce788edb101f/propSheetGroup/propSheets/b00c0b1c-d007-4c19-a9dd-2e1397df8f1
f",
    "version": 3,
    "versionCount": 3,
    "commit": 111,
    "versioned": true
  },
  "extendedSecurity": {
    "read": true,
    "write": true,
    "Create Resources": true,
    "Edit Resources": true,
    "View Resources": true,
    "teams": [
    ]
  }
}
```

Related REST command: Get information about a resource.

getResourceInventoryEntriesForComponent

Get current resource inventory entries for a component

This command returns a JSON array of current resource inventory entries for a component deployed to a resource. If there are no versions of the component deployed, the array will be empty. If there is one or more versions deployed, typically the first entry will correspond to a full version and subsequent entries will correspond to incremental versions. The entries are in ascending order by the date the entry was created, which will be when the deploying component process completed.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceInventoryEntriesForComponent [parameters]
```

Parameters

Table 970. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
component	string	true	ID or name of the component

Related REST command: Get current resource inventory entries for a component.

getResourceProperties

List resource properties

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceProperties [parameters]
```

Parameters

Table 971. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResourceProperties
  -resource "/My resource"
```

Example

Example response

```
[
  {
    "id": "bd20924e-8547-4d41-b9b1-36b41278d224",
    "name": "property1",
    "value": "1",
    "description": "A new property",
    "secure": false
  },
  {
    "id": "18963e9e-2cca-47ac-807a-7fc5e67c49ee",
    "name": "property2",
    "value": "",
    "description": "A property with no value",
    "secure": false
  },
  {
    "id": "9cc56712-65f8-41fe-8fd6-d07414364c98",
    "name": "property3",
```

```
    "value": "****",
    "description": "A secure property",
    "secure": true
  }
]
```

Related REST command: List resource properties.

getResourceProperty

Get a resource property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceProperty [parameters]
```

Parameters

Table 972. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResourceProperty
  -resource "/My resource"
  -name property1
```

Related REST command: Get a resource property.

getResourceRoleProperties

List resource role properties

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceRoleProperties [parameters]
```

Parameters

Table 973. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Related REST command: List resource role properties.

getResourceRoleProperty

Get a resource role property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceRoleProperty [parameters]
```

Parameters

Table 974. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
name	string	true	Name of the property

Related REST command: Get a resource role property.

getResourceTemplate

Get a resource template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceTemplate [parameters]
```

Parameters

Table 975. Parameters

Parameter	Type	Required	Description
template	string	true	Name or ID of resource template

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResourceTemplate
  -template MyResourceTemplate
```

Example

Example response


```
{
  "propDefs": [],
  "security": {
    "read": true,
    "Create Resource Templates": true,
    "Delete": true,
    "Edit Basic Settings": true,
    "Manage Resources": true,
    "Manage Teams": true,
    "View Resource Templates": true
  },
  "propSheetDef": {
    "id": "27c709ec-07d8-49b2-bf20-639b07e625d5",
    "versioned": false,
    "resolveHttpValuesUrl": "property/propSheetDef/27c709ec-07d8-49b2-bf20-639b07e625d5/resolveHttpValues"
  },
  "propSheet": {
    "id": "4d0b0d8c-8405-44de-a8a9-4fe7ffb36c19",
    "versioned": false
  },
  "extendedSecurity": {
    "read": true,
    "Create Resource Templates": true,
    "Delete": true,
    "Edit Basic Settings": true,
    "Manage Resources": true,
    "Manage Teams": true,
    "View Resource Templates": true,
```

```
    "teams": []
  },
  "id": "95583b93-13e3-484c-b444-480008e5083d",
  "securityResourceId": "ed438378-f215-4fd0-92d0-1cfcce26e621",
  "name": "MyResourceTemplate",
  "description": "",
  "deleted": false
}
```

Related REST command: Get a resource template.

getResourceTemplateChildren

Get all children or descendants of a resource template

Results include the base resource and resources inherited from ancestors of this template.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourceTemplateChildren [parameters]
```

Parameters

Table 976. Parameters

Parameter	Type	Required	Description
template	string	true	Name or ID of the resource template. If this is a blueprint identified by name rather than ID, a parent application must also be specified using the -application flag.
application	string	false	Name or ID of the parent application. Required if the template represents an application blueprint identified by name.
recursive	boolean	false	Get all descendants of the template (not just immediate children). Default is false.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResourceTemplateChildren
  -template MyResourceTemplate
```

Related REST command: Get all children or descendants of a resource template.

getResources

List resources

This command returns a JSONArray representation of all top-level resources, or all children of a specified parent resource.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResources [parameters]
```

Parameters

Table 977. Parameters

Parameter	Type	Required	Description
parent	string	false	Specify the path to a parent resource to show children of that resource, such as /AllAgents/MyAgentGroup. Leave this field blank to show all top-level resources. (Optional)
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
includeInventory	boolean	false	

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResources
  -parent "/JPetStore agents"
```

Example

Example response

```
[
  {
    "id": "5fcc9f43-63eb-40bc-bcd6-3c0b2c124dce",
    "securityResourceId": "67bd54e9-39c5-4f49-b4d6-cf6eee286589",
    "name": "agent1",
    "path": "\/JPetStore agents\/agent1",
    "active": true,
    "description": "",
    "inheritTeam": true,
    "impersonationPassword": "****",
    "impersonationUseSudo": false,
    "impersonationForce": false,
    "type": "agent",
    "status": "ONLINE",
```

```
    "hasAgent": true,
    "tags": []
  },
  {
    "id": "1fd3d080-df81-4cd2-aa6d-58ae551fc3213",
    "securityResourceId": "38a18720-abef-4931-9ac0-62d19c06e506",
    "name": "agent2",
    "path": "\/JPetStore agents\/agent2",
    "active": true,
    "description": "",
    "inheritTeam": true,
    "impersonationPassword": "****",
    "impersonationUseSudo": false,
    "impersonationForce": false,
    "type": "agent",
    "status": "OFFLINE",
    "hasAgent": true,
    "tags": []
  }
]
```


Related REST command: List resources.

getResourcesInRecycleBin

List resources in the recycle bin.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourcesInRecycleBin [parameters]
```

Parameters

Table 978. Parameters

Parameter	Type	Required	Description
rowsPerPage	int	false	Maximum number of resource to return. Default 10.
pageNumber	int	false	1-Indexed page number. Default 1.

Related REST command: List resources in the recycle bin..

getResourcesWithComponentVersion

List resources that have a specified component version

This command returns a JSONArray representation of all resources that have the specified component version installed.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getResourcesWithComponentVersion [parameters]
```

Parameters

Table 979. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
version	string	true	Version name

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getResourcesWithComponentVersion
  -component "JPetStore-APP"
  -version "1.0"
```

Example

Example response

```
[
  {
    "id": "53b4b54f-5b3f-4fef-91dd-dea8ffe15dc3",
    "name": "JPetStore-APP",
    "path": "/JPetStore agents/vm192-168/JPetStore-APP",
    "active": true,
    "description": "",
    "inheritTeam": true,
    "impersonationPassword": "****",
    "impersonationUseSudo": false,
    "impersonationForce": false,
    "type": "subresource",
    "status": "ONLINE",
    "hasAgent": true,
    "tags": [
    ]
  }
]
```

Related REST command: List resources that have a specified component version.

getSecretStore

Get information about a secret store.

This command returns a JSON representation of a secret store.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSecretStore [parameters]
```

Parameters

Table 980. Parameters

Parameter	Type	Required	Description
store	string	true	Name or id of the secret store.

Related REST command: Get information about a secret store..

getSecretStores

Get information about all secret stores.

This command returns a JSON representation of all secret stores.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSecretStores
```

Parameters

None.

Related REST command: Get information about all secret stores..

getServerKeystoreImportTask

Get a keystore import task

Returns a representation of a keystore import task.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getServerKeystoreImportTask [parameters]
```

Parameters

Table 981. Parameters

Parameter	Type	Required	Description
task	string	true	ID of an import task.

Related REST command: Get a keystore import task.

getServerKeystoreInfo

Get information about the server keystore

Returns a representation of the content of the server keystore, which contains key pairs and certificates used for agent communication. This keystore is the file at path appdata/conf/server.keystore.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getServerKeystoreInfo
```

Parameters

None.

Related REST command: Get information about the server keystore.

getSnapshot

Get information about a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSnapshot [parameters]
```

Parameters

Table 982. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getSnapshot
  -application JPetStore
  -snapshot mySnapshot
```

Example

Example response


```
{
  "application": {
    "security": {
      "read": true,
      "Create Applications": true,
      "Delete": true,
      "Edit Basic Settings": true,
      "Manage Blueprints": true,
      "Manage Components": true,
      "Manage Environments": true,
      "Manage Processes": true,
      "Manage Properties": true,
      "Manage Snapshots": true,
      "Manage Teams": true,
      "Run Component Processes": true,
      "View Applications": true
    },
    "componentCount": 16,
    "extendedSecurity": {
      "read": true,
      "Create Applications": true,
      "Delete": true,
      "Edit Basic Settings": true,
      "Manage Blueprints": true,
      "Manage Components": true,
      "Manage Environments": true,
      "Manage Processes": true,
      "Manage Properties": true,
      "Manage Snapshots": true,
      "Manage Teams": true,
      "Run Component Processes": true,
      "View Applications": true,
      "teams": [{
        "teamId": "86739325-1ebb-4fb9-8a0f-a73c1e4d88c8",
        "teamLabel": "Landscaper",
        "resourceTypeId": "200000000000000000000000000000100",
        "resourceTypeName": "Application"
      }]
    },
    "propSheet": {
      "id": "ad6279a0-f140-4398-a238-cedb26880e2c",
      "path": "applications\/413b1470-3dae-446c-95fa-77873a5daaa9\/propSheet",
      "version": 1,
      "versionCount": 1,
      "commit": 10231,
      "versioned": true
    },
    "id": "413b1470-3dae-446c-95fa-77873a5daaa9",
    "securityResourceId": "ba1e7e3e-8642-45f9-a378-7528a82deb9a",
    "name": "JPetStore",
    "description": "",
    "created": 1389979285087,
    "enforceCompleteSnapshots": false,
    "active": true,
    "tags": [],
    "user": "Michael (mdelder@us.ibm.com)"
```

```
  },
  "id": "8724200d-9cd1-4b31-90e2-9ab8b0edb55d",
  "name": "MySnapshot",
  "description": "",
  "created": 1410462609512,
  "active": true,
  "versionsLocked": false,
  "configLocked": false,
  "applicationId": "413b1470-3dae-446c-95fa-77873a5daaa9",
  "user": "jsmith"
}
```

Related REST command: Get information about a snapshot.

getSnapshotConfiguration

Get the list of configuration objects in a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSnapshotConfiguration [parameters]
```

Parameters

Table 983. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Related REST command: Get the list of configuration objects in a snapshot.

getSnapshotVersions

Get the list of versions in a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSnapshotVersions [parameters]
```

Parameters

Table 984. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot

Table 984. Parameters (continued)

Parameter	Type	Required	Description
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getSnapshotVersions
  -application JPetStore
  -snapshot mySnapshot
```

Related REST command: Get the list of versions in a snapshot.

getSnapshotsInApplication

Get information about snapshots in an application

Snapshots are listed from most recent to least recent.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSnapshotsInApplication [parameters]
```

Parameters

Table 985. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
maxResults	int	false	Cap the number of results. Any value less than 1 returns all results.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getSnapshotsInApplication
  -application MyApplication
```

Example

Example JSON request

```
[
  {
    "id": "305749d0-9838-4fce-a8bd-7d25b4ffd5f7",
    "name": "Snapshot1",
    "created": 1441386527766,
    "active": true,
    "versionsLocked": false,
    "configLocked": false,
    "applicationId": "2d741420-fd31-443d-95eb-472d334f8ec9",
    "user": "jsmith"
  },
  {
    "id": "5ab250dd-37f8-4a6e-9ec1-8cf0b547eb25",
    "name": "Snapshot2",
    "created": 1441386081196,
    "active": true,
    "versionsLocked": false,
    "configLocked": false,
    "applicationId": "2d741420-fd31-443d-95eb-472d334f8ec9",
    "user": "jsmith"
  }
]
```

Related REST command: Get information about snapshots in an application.

getStatus

Get information about a status

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getStatus [parameters]
```

Parameters

Table 986. Parameters

Parameter	Type	Required	Description
type	string	false	Type of status (inventory, version, or snapshot); this parameter is required you specify the name of the status instead of the ID
status	string	true	Name or ID of the status

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
```

```
  getStatus
  -type inventory
  -status Active
```

Example

Example response

```
[
  {
    "id": "061056cc-9433-4036-ac11-b3c0b84e7018",
    "type": "INVENTORY",
    "name": "Active",
    "active": true,
    "description": "Currently Deployed",
    "color": "#B3D66D",
    "unique": true
  }
]
```

Related REST command: Get information about a status.

getStatusList

Get a list of statuses on a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getStatusList [parameters]
```

Parameters

Table 987. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot.
application	string	false	Name or ID of the application; this value is required if you specify the snapshot name instead of ID.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getStatusList
  -application MyApplication
  -snapshot MySnapshot
```

Example

Example response

```
[
  {
    "id": "5ab6cc4f-393f-4fa2-ac35-d30e282648c4",
    "type": "SNAPSHOT",
    "name": "Good",
    "description": "",
    "deleted": false,
    "color": "#FFCF01",
    "unique": false,
    "created": 1450130997618
  },
  {
    "id": "51259a1f-69bc-4a3a-b1f4-c1d391b7794a",
    "type": "SNAPSHOT",
    "name": "Interesting",
    "description": "",
    "deleted": false,
    "color": "#007670",
    "unique": false,
    "created": 1450131001305
  }
]
```

Related REST command: Get a list of statuses on a snapshot.

getStatuses

List statuses

This command lists all the statuses that are associated with a specified status type.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getStatuses [parameters]
```

Parameters

Table 988. Parameters

Parameter	Type	Required	Description
type	string	true	Status type (inventory, version, or snapshot)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
```

```
  getStatuses
  -type inventory
```

Example

Example response

```
[
  {
    "id": "061056cc-9433-4036-ac11-b3c0b84e7018",
    "type": "INVENTORY",
    "name": "Active",
    "active": true,
    "description": "Currently Deployed",
    "color": "#B3D66D",
    "unique": true
  }
]
```

Related REST command: List statuses.

getSystemConfiguration

List system configuration properties

These properties are available on the System Settings page of the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSystemConfiguration
```

Parameters

None.

Example

Example response

```
{
    "vendorName": "",
    "externalURL": "https://example.hostname.com:8443",
    "externalUserURL": "https://example.hostname.com:8443",
    "repoAutoIntegrationPeriod": 300,
    "deployMailHost": "smtp.example.com",
    "deployMailPassword": "****",
    "deployMailPort": 25,
    "deployMailSecure": false,
    "deployMailSender": "sender@example.com",
    "deployMailUsername": "username",
    "snapshotDaysToKeep": -1,
    "cleanupHourOfDay": 0,
    "cleanupDaysToKeep": -1,
    "cleanupCountToKeep": -1,
    "environmentDaysToKeep": -1,
    "environmentCountToKeep": -1,
```

```
    "cleanupArchivePath": "",
    "historyCleanupHour": 0,
    "historyCleanupMinute": 0,
    "historyCleanupDaysToKeep": 730,
    "historyCleanupDuration": 23,
    "historyCleanupEnabled": false,
    "auditLogReadEntriesEnabled": false,
    "auditLogCleanupEnabled": false,
    "auditLogCleanupHour": 0,
    "auditLogCleanupMinute": 0,
    "auditLogRetentionLength": 30,
    "enableInactiveLinks": false,
    "enablePromptOnUse": true,
    "enableAllowFailure": false,
    "validateAgentIp": false,
    "skipCollectPropertiesForExistingAgents": false,
    "minimumPasswordLength": 1,
    "requireUppercase": false,
    "requireLowercase": false,
    "requireSpecials": false,
    "requireDigits": false,
    "requireDuplicate": false,
    "enableUIDebugging": true,
    "messageOfTheDay": "",
    "isCreateDefaultChildren": false,
    "requireCommentForProcessChanges": false,
    "requireProcessPromotionApproval": false,
    "failProcessesWithUnresolvedProperties": false,
    "enforceDeployedVersionIntegrity": false,
    "copyToCodestation": true,
    "discoveryExpiry": 300,
    "configureExpiry": 43200,
    "importExpiry": 7200,
    "pluginStepExpiry": 86400,
    "useDefaultATRIfNotSpecified": false,
    "safeEditEnabled": false,
    "allowProcessLocking": false,
    "requireProcessLocking": false,
    "maintenanceModeEnabled": false,
    "keepMeLoggedInHours": 12,
    "versionImportKeepRecords": 100,
    "licenseServerUrl": "use the default that the server tooltip recommends",
    "licenseMode": "MIXED",
    "agentAutoLicenseType": "",
    "defaultLocale": "en_US",
    "defaultSnapshotLockType": "ALL",
    "envCompPropsOverrideEnvProps": false,
    "deleteEnvResources": false,
    "vcCompressionUpgradeEnabled": false,
    "vcCompressionUpgradeCompleted": false,
    "trustedProxyIPs": ""
}
```


Related REST command: List system configuration properties.

getSystemProperties

List system properties

This command returns a JSONArray representation of the system properties, which behave like global variables for the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSystemProperties
```

Parameters

None.

Example

Example response

```
[
  {
    "id": "af25c8b2-a69a-4d1b-a0ed-ddb226dfac18",
    "name": "System property 1",
    "value": "value001",
    "description": "This is a system property",
    "secure": false
  }
]
```

Related REST command: List system properties.

getSystemProperty

Get the value of a system property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getSystemProperty [parameters]
```

Parameters

Table 989. Parameters

Parameter	Type	Required	Description
name	string	true	Name of the property

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getSystemProperty
  -name property1
```

Example

Example response

```
value001
```

Related REST command: Get the value of a system property.

getTagsForType

Get the Tags for an object type.

This command returns a JSON array containing all active tags for the given object type.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTagsForType [parameters]
```

Parameters

Table 990. Parameters

Parameter	Type	Required	Description
objectType	string	true	Object type for the tags. e.g. Agent, Application, ApplicationTemplate, Component, ComponentTemplate, Resource

Related REST command: Get the Tags for an object type..

getTagsOnAgent

List all tags on an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTagsOnAgent [parameters]
```

Parameters

Table 991. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getTagsOnAgent
  -agent Agent1
```

Example

Example response

```
[
  {
    "id": "34afc9d1-e6d7-420f-a40c-30e6d3bf0c34",
    "name": "Tag1",
    "color": "#FFFFFF",
    "description": "",
    "objectType": "Agent"
  },
  {
    "id": "74c02e58-e532-47d7-ba98-66d1ac20c361",
    "name": "Tag2",
    "color": "#7fff00",
    "description": "",
    "objectType": "Agent"
  }
]
```

Related REST command: List all tags on an agent.

getTagsOnApplication

List all tags on an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTagsOnApplication [parameters]
```

Parameters

Table 992. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getTagsOnApplication
  -application JPetStore
```

Example

Example response

```
[
  {
    "id": "f3c52993-ffe4-464a-b34c-0abc1df7f995",
    "name": "Tag1",
    "color": "#dda0dd",
    "description": "",
    "objectType": "Application"
  },
  {
    "id": "ae47b2dc-8153-42c9-8094-207703a439e4",
    "name": "Tag2",
    "color": "#ffff00",
    "description": "",
    "objectType": "Application"
  }
]
```

Related REST command: List all tags on an application.

getTagsOnComponent

Return all tags on a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTagsOnComponent [parameters]
```

Parameters

Table 993. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getTagsOnComponent
  -component JPetStore-APP
```

Example

Example response

```
[
  {
    "id": "13a1604a-ecea-4f07-b85e-6b395ade161a",
    "name": "Tag1",
    "color": "#FFFFFF",
    "description": "",
    "objectType": "Component"
  },
  {
    "id": "d140ae5a-dfb0-477a-910b-762cc0d99cc6",
    "name": "Tag2",
    "color": "#7fff00",
    "description": "",
    "objectType": "Component"
  }
]
```

Related REST command: Return all tags on a component.

getTagsOnResource

List the tags on a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTagsOnResource [parameters]
```

Parameters

Table 994. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getTagsOnResource
  -resource /Resource1/Subresource1
```

Example

Example response

```
[
  {
    "id": "eeff1b7d-ef7c-4052-812b-a97405b97606",
    "name": "Tag1",
    "color": "#FFFFFF",
    "description": "",
    "objectType": "Resource"
  },
  {
    "id": "b03313f2-a3fa-4ac8-a84a-32869958ba06",
    "name": "Tag2",
    "color": "#87ceeb",
    "description": "",
    "objectType": "Resource"
  }
]
```

Related REST command: List the tags on a resource.

getTasksForUser

Return all approval tasks and manual tasks assigned to the user

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTasksForUser
```

Parameters

None.

Related REST command: Return all approval tasks and manual tasks assigned to the user.

getTeam

List information about a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getTeam [parameters]
```

Parameters

Table 995. Parameters

Parameter	Type	Required	Description
team	string	true	Name of the team

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getTeam
  -team MyNewTeam
```

Example

Example response

```
{
  "id": "769c2486-ed40-4a3a-ac52-faa33d30cfd5",
  "name": "MyNewTeam",
  "isDeletable": true,
  "roleMappings": [
    {
      "user": {
        "id": "20000000000000000000000001000000",
        "name": "admin",
        "displayName": "admin",
        "deleted": false,
        "deletedDate": 0,
        "authenticationRealm": "20000000000000000000000000000001",
        "isLockedOut": false,
        "isDeletable": true
      },
      "role": {
        "id": "20000000000000000000000000010001",
        "name": "Administrator",
```

```
        "description": "",
        "isDeletable": true
      }
    },
    {
      "user": {
        "id": "2d2c8241-ba69-4b63-abf0-de90b7d4ceeb",
        "name": "jdoe",
        "actualName": "Jane Doe",
        "displayName": "Jane Doe (jdoe)",
        "email": "",
        "deleted": false,
        "deletedDate": 0,
        "authenticationRealm": "20000000000000000000000000000001",
        "isLockedOut": false,
        "isDeletable": true
      },
      "role": {
        "id": "90f5f9a0-3d01-4aaa-a276-74214188fd1d",
        "name": "Developer",
        "isDeletable": true
      }
    },
    {
      "user": {
        "id": "b1e20a57-2897-4372-b6c6-662c719e6dda",
        "name": "jsmith",
        "actualName": "Joe Smith",
        "displayName": "Joe Smith (jsmith)",
        "email": "",
        "deleted": false,
        "deletedDate": 0,
        "authenticationRealm": "20000000000000000000000000000001",
        "isLockedOut": false,
        "isDeletable": true
      },
      "role": {
        "id": "e407437d-0c5f-446f-939f-9fa4d7df4d5e",
        "name": "Observer",
        "isDeletable": true
      }
    }
  ]
}
```


Related REST command: List information about a team.

getUser

List information about a user

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getUser [parameters]
```

Parameters

Table 996. Parameters

Parameter	Type	Required	Description
user	string	true	Name/ID of the user. The name will be case-sensitive if a case-sensitive realm is provided.
realm	string	false	Limit any user name search to given AuthenticationRealm.

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getUser
  -user jsmith
```

Example

Example response

```
{
  "id": "2507eaaf-c98e-46ee-936d-bc538bee9dc4",
  "name": "jsmith",
  "actualName": "Jane C. Smith",
  "displayName": "Jane C. Smith (jsmith)",
  "email": "jsmith@example.org",
  "deleted": false,
  "deletedDate": 0,
  "authenticationRealm": "20000000000000000000000000000001",
  "isLockedOut": false,
  "isDeletable": true,
  "groups": [
  ]
}
```

Related REST command: List information about a user.

getUserByEmail

Get the user or users with a given e-mail address

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getUserByEmail [parameters]
```

Parameters

Table 997. Parameters

Parameter	Type	Required	Description
email	string	true	E-mail address of the user
allUsers	boolean	false	Specify true to retrieve all users with the e-mail address or false to return only the first user that matches the address

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getUserByEmail
  -email jsmith@us.ibm.com
  -allUsers true
```

Example

Example response

```
[{"id":"bade154b-c8c9-4aa6-a875-4b56d7132be5","name":"jsmith"},
{"id":"2c1c9012-2111-4d6a-b5f0-b6cabff94713","name":"janesmith"}]
```

Related REST command: Get the user or users with a given e-mail address.

getUsersThatCanRunDeploymentTriggers

Get all users that can run deployment triggers.

This command returns a JSON representation of all users that can run deployment triggers.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getUsersThatCanRunDeploymentTriggers
```

Parameters

Table 998. Parameters

Parameter	Type	Required	Description
environment	string	true	ID of the environment.

Related REST command: Get all users that can run deployment triggers..

getVaultAppRole

Get information about a Vault AppRole

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultAppRole [parameters]
```

Parameters

Table 999. Parameters

Parameter	Type	Required	Description
store	string	false	Store ID/Name
vaultAppRole	string	true	Name or ID of the Vault App Role

Related REST command: Get information about a Vault AppRole.

getVaultAppRoles

List the Vault AppRoles for a Vault store

This command returns a list of app roles within the given Vault store represented as a JSON Array.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultAppRoles [parameters]
```

Parameters

Table 1000. Parameters

Parameter	Type	Required	Description
store	string	true	Name or ID of the Vault Store to list AppRoles for.

Related REST command: List the Vault AppRoles for a Vault store.

getVaultCertificate

Get information about a Vault Certificate

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultCertificate [parameters]
```

Parameters

Table 1001. Parameters

Parameter	Type	Required	Description
store	string	false	Store ID/Name
vaultCertificate	string	true	Name or ID of the Vault Certificate

Related REST command: Get information about a Vault Certificate.

getVaultCertificates

List the Vault Certificates for a Vault store

This command returns a list of certificates within the given Vault store represented as a JSON Array.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultCertificates [parameters]
```

Parameters

Table 1002. Parameters

Parameter	Type	Required	Description
store	string	true	Name or ID of the Vault Store to list Certificates for.

Related REST command: List the Vault Certificates for a Vault store.

getVaultLdapUser

Get information about a Vault Ldap User

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultLdapUser [parameters]
```

Parameters

Table 1003. Parameters

Parameter	Type	Required	Description
store	string	false	Store ID/Name
vaultLdapUser	string	true	Name or ID of the Vault Ldap User

Related REST command: Get information about a Vault Ldap User.

getVaultLdapUsers

List the Vault Ldap Users for a Vault Secret Store

This command returns a list of ldap users within the given Vault store represented as a JSON Array.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVaultLdapUsers [parameters]
```

Parameters

Table 1004. Parameters

Parameter	Type	Required	Description
store	string	true	Name or Id of the Vault Store to list Ldap Users for.

Related REST command: List the Vault Ldap Users for a Vault Secret Store.

getVersionId

Get a version ID

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVersionId [parameters]
```

Parameters

Table 1005. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getVersionId
  -component MyComponent
  -version 1.2
```

Example

Example response

```
1a53a4a6-bfcd-4014-a0ab-83f97077a4d4
```

Related REST command: Get a version ID.

getVersionLink

Get a link on a component version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVersionLink [parameters]
```

Parameters

Table 1006. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version

Table 1006. Parameters (continued)

Parameter	Type	Required	Description
linkName	string	true	Name of the link

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getVersionLink
  -component JPetStore-APP
  -version 1.0
  -linkName "Link to MyCompany example web site"
```

Example

Example response

```
{
  "name": "Link to foo web site",
  "value": "http://www.foo.com"
}
```

Related REST command: Get a link on a component version.

getVersionLinks

List links on a version

This command lists all links that are associated with a specified component version.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVersionLinks [parameters]
```

Parameters

Table 1007. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  getVersionLinks
  -component JPetStore-APP
  -version 1.0
```

Example

Example response

```
[
  {
    "name": "Link to MyCompany example web site",
    "value": "http://mycompany.example.com"
  },
  {
    "name": "Link to foo web site",
    "value": "http://www.foo.com"
  }
]
```

Related REST command: List links on a version.

getVersionProperties

List the properties on a component version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getVersionProperties [parameters]
```

Parameters

Table 1008. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
```

```
  getVersionProperties
  -component JPetStore-APP
  -version 1.0
```

Example

Example response

```
[
  {
    "id": "33528ab5-a7be-46cb-9f0c-916a42bd2f62",
    "name": "Prop1",
    "value": "value1",
    "description": "",
    "secure": false
  },
  {
    "id": "92339074-ab59-4fe3-936f-9ceec16ce5e6",
    "name": "Prop2",
    "value": "value1",
    "description": "",
    "secure": false
  }
]
```

Related REST command: List the properties on a component version.

getWebhookTemplate

Get information about a webhook template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getWebhookTemplate [parameters]
```

Parameters

Table 1009. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	Name or ID of the webhook template to retrieve

Related REST command: Get information about a webhook template.

getWebhookTemplates

List the viewable webhook templates

This command returns a list of webhook templates represented as a JSON Array.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getWebhookTemplates
```

Parameters

None.

Related REST command: List the viewable webhook templates.

getWorkflowTraceLogs

Returns full workflow trace with execution logs as a zip file in the current directory.If the underlying workflow is not complete, an HTTP 412 response is returned.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getWorkflowTraceLogs [parameters]
```

Parameters

Table 1010. Parameters

Parameter	Type	Required	Description
workflow	string	true	ID of the workflow.

Related REST command: Get full workflow trace with execution logs as a zip file in the current directory.

getZOSVersionCumulativeManifest

Get the cumulative manifest for a linked z/OS version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getZOSVersionCumulativeManifest [parameters]
```

Parameters

Table 1011. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID

Table 1011. Parameters (continued)

Parameter	Type	Required	Description
version	string	true	Name or ID of the version

Related REST command: Get the cumulative manifest for a linked z/OS version.

getZOSVersionDeployManifest

Get a deploy manifest for deploying a version into an environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getZOSVersionDeployManifest [parameters]
```

Parameters

Table 1012. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID.
version	string	true	Name or ID of the version for which to generate the deploy manifest.
fromVersion	string	false	Name or ID of the current version from which to generate the deploy manifest.

Related REST command: Get a deploy manifest for deploying a version into an environment..

getZOSVersionsWithCommitId

Get the versions from a component with a given commit ID.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  getZOSVersionsWithCommitId [parameters]
```

Parameters

Table 1013. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component.
commitId	string	true	Commit ID of the version.
parentCommitId	string	false	Commit ID of the parent version.

Related REST command: Get the versions from a component with a given commit ID..

importApplication

Import application from json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importApplication [parameters]
```

Parameters

Table 1014. Parameters

Parameter	Type	Required	Description
componentUpgradeType	string	false	Supported values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
applicationUpgradeType	string	false	Supported values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
processUpgradeType	string	false	Supported values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
resourceTemplateUpgradeType	string	false	Supported values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.

Table 1014. Parameters (continued)

Parameter	Type	Required	Description
importFullResourceTrees	boolean	false	Whether to import full resource tree or just base resources. true or false. Default false
disposition	string	true	create or update.

Related REST command: Import application from json file..

importApplicationProcess

Import an application process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importApplicationProcess [parameters]
```

Parameters

Table 1015. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
processUpgradeType	string	false	Supported values are, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, FAIL_IF_EXISTS, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS.

Example

Command to import application process from a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
      importApplicationProcess
      -application JPetStore
      application-process-JPetStoreprocess.json
```

Example

Command to print available options

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
      importApplicationProcess -h
```

Example

Example response

```
{
  "id": "187b908b-570f-1121-b943-e058185063d5",
  "name": "Deploy JPetStore",
  "description": "",
  "active": true,
  "inventoryManagementType": "AUTOMATIC",
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
  "versionCount": 4,
  "version": 4,
  "commit": 963,
  "path":
 "applications\/187b908b-3bc1-0bcb-0233-abff5fa79d79\/processes\/187b908b-570f-1121-b943-e058185063d5",
  "deleted": false,
  "metadataType": "applicationProcess",
  "disableSnapshots": false
}
```

Related REST command: Import an application process.

importComponent

Import component from json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importComponent [parameters]
```

Parameters

Table 1016. Parameters

Parameter	Type	Required	Description
componentTemplateUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
processUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.

Table 1016. Parameters (continued)

Parameter	Type	Required	Description
disposition	string	true	create or update

Related REST command: Import component from json file..

importComponentProcess

Import a component process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importComponentProcess [parameters]
```

Parameters

Table 1017. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
processUpgradeType	string	false	Supported values are, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, FAIL_IF_EXISTS, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS.

Example

Command to import component process from a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
      importComponentProcess
      -component JPetStore-APP
      component-process-JPetStorecomponent.json
```

Example

Example response

```
{
  "id": "187b908b-570f-1121-b943-e058185063d5",
  "name": "JPetStore-APP",
  "description": "",
  "active": true,
  "inventoryManagementType": "AUTOMATIC",
```

```
  "offlineAgentHandling": "PRE_EXECUTION_CHECK",
  "versionCount": 4,
  "version": 4,
  "commit": 963,
  "path": "components\/187b908b-3bc1-0bcb-0233-abff5fa79d79\/processes\/187b908b-570f-1121-b943-e058185063d5",
  "deleted": false,
  "metadataType": "componentProcess",
  "disableSnapshots": false
}
```

Example

Command to print available options

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
      importComponentProcess -h
```

Related REST command: Import a component process.

importComponentTemplate

Import component template from json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importComponentTemplate [parameters]
```

Parameters

Table 1018. Parameters

Parameter	Type	Required	Description
processUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
disposition	string	true	create or update.

Example

Command to import a component template process from a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  importComponentTemplate
  -componentTemplate componentTemplates
  -processUpgradeType FAIL_IF_DOESNT_EXIST
  -disposition update
  ./component-process-file.json
```

Example

Response

```
{
  "id": "1849bc16-e5e8-2705-58b1-33e021705448",
  "securityResourceId": "1849bc16-e5b8-0d7f-b050-473e9b11e726",
  "versionCount": 28,
  "version": 28,
  "path": "componentTemplates\/1849bc16-e5e8-2705-58b1-33e021705448",
  "name": "componentTemplate",
  "description": "",
  "ignoreQualifiers": 2,
  "componentType": "ZOS",
  "sourceConfigPluginName": "Git",
  "tags": [],
  "active": true,
  "created": 1669060456020,
  "user": "jsmith",
  "importConfigurationStrategy": "default"
}
```

Related REST command: Import component template from json file..

importComponentTemplateProcess

Import a component template process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importComponentTemplateProcess [parameters]
```

Parameters

Table 1019. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template
processUpgradeType	string	false	Supported values are, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, FAIL_IF_EXISTS, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS.

Related REST command: Import a component template process.

importGenericProcess

Import process from json file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importGenericProcess [parameters]
```

Parameters

Table 1020. Parameters

Parameter	Type	Required	Description
disposition	string	true	create or update

Related REST command: Import process from json file..

importGroup

Import a group

Specify the details about the group in the JSON file. You can list one or more members of the new group in the 'users' array, or you can leave the array blank to create an empty group. Groups can be created only in Internal Storage authorization realms.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importGroup [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "authorizationRealm": "Authorization realm name",
  "description": "Group description. Optional.",
  "name": "Group name",
  "users": ["User names"]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  importGroup newGroup.json
```

Example

Example JSON request

```
{
  "name": "group1",
  "authorizationRealm": "Internal Security",
  "users": [
    "jdoe",
    "jsmith"
  ]
}
```

Example

Example response

```
{
  "id": "9dc44e77-e241-480e-8fbb-50f84b64b24b",
  "name": "group1",
  "enabled": true
}
```

Related REST command: Import a group.

importLdapUser

Import a user from LDAP

This command adds one or more users from an LDAP realm to the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importLdapUser [parameters]
```

Parameters

Table 1021. Parameters

Parameter	Type	Required	Description
user	string	true	Name or pattern of the users to be imported. Supports wildcard character *.
authenticationRealm	string	true	Authentication realm name or ID

Example

Example of importing a user with auth token

```
curl -sku 'PasswordIsAuthToken:23453425-dfgfgd-23432-sdfsf'
  'https://deployserver.example.com:8443/cli/user/import?user=jsmith
  &authenticationRealm=Internal Security'
  -X PUT
```

Example

Example of importing users using a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
      importLdapUser importUsers.json
```

Example

Example JSON request

```
{
        "name": "importUsers",
        "authorizationRealm": "Internal Security",
        "users": [
                  "jdoe",
                  "jsmith"
                 ]
}
```

Example

Example response

```
{
  "id":"18b3e4ac-13d2-9ecb-f12c-d078fe0f06e5",
  "name":"jdoe",
  "actualName":"John Doe",
  "displayName":"JOHN DOE",
  "email":"john.doe@example.org",
  "deleted":false,
  "deletedDate":0,
  "authenticationRealm":"18b3e49e-987d-6595-7a67-8b6d47c8cbf8",
  "isLockedOut":false,
  "lastLoginDate":1697793306115,
  "isDeletable":true,
  "namePattern":"john.doe@example.org"
}
```

Related REST command: Import a user from LDAP.

importSnapshot

Import snapshot from zip file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importSnapshot [parameters]
```

Parameters

Table 1022. Parameters

Parameter	Type	Required	Description
componentUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
applicationUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
processUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
resourceTemplateUpgradeType	string	false	Suppport values are UPGRADE_IF_EXISTS, CREATE_NEW_IF_EXISTS, FAIL_IF_DOESNT_EXIST, UPGRADE_IF_EXISTS, USE_EXISTING_IF_EXISTS. Default is UPGRADE_IF_EXISTS.
importFullResourceTrees	boolean	false	Whether to import full resource tree or just base resources. true or false. Default false
disposition	string	true	create or update.

Related REST command: Import snapshot from zip file..

importVersions

Import new component versions

This command polls the source configuration for new component versions. Add property name/value pairs as necessary - this can be used to add specific versions.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  importVersions [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Component name or ID",
  "properties": {"Source configuration plugin property
  name": "Property value"}
```

```
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  importVersions newVersions.json
```

Example

Example JSON request

```
{
  "component": "JPetStore-WEB",
  "properties": {
    "Prop1": "value1"
  }
}
```

Related REST command: Import new component versions.

installAgent

Install an agent

This command installs an agent over SSH.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  installAgent [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "host": "Host name or IP address of the system to
  install the agent on",
  "installDir": "Installation directory for the agent",
  "javaHomePath": "Path to Java on the agent system",
  "name": "Name of the agent resource",
  "port": "SSH port of the agent system; the default port
  is 22",
  "sshPassword": "SSH password for the agent system; leave
  blank to use public key authentication instead",
  "sshUsername": "SSH user name to use to connect to the
  agent system",
  "tempDirPath": "Path to directory to install from on the
  agent system",
  "webProxyUri": "Web socket URI for the relay server (e.
```

```
  g. https://relay:20080).",
  "webUri": "Web socket URI for the Deploy server (e.g.
  wss://server:7919)."
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  installAgent
  -name NewAgent
  -host agenthost.example.com
  -port 22
  -sshUsername root
  -sshPassword 12345
  -installDir /opt/devops-deploy/agent
  -javaHomePath /usr/lib/java
  -tempDirPath /tmp
  -serverHost host.example.com
  -serverPort 7918
```

Related REST command: Install an agent.

installAgentJava

Install a Java package as a new managed Java installation

This command requests the agent install Java, but does not wait for or report the result. The returned maintenance task can be used to query the result.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  installAgentJava [parameters]
```

Parameters

Table 1023. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
package	string	true	Name or ID of the Java package

Related REST command: Install a Java package as a new managed Java installation.

installLocalPlugin

Installs a plugin into the server from a local zip file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  installLocalPlugin
```

Parameters

None.

Related REST command: Install a local plug-in.

installPlugin

Installs a plugin into the server from a remote location defined in the plugin manifest.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  installPlugin [parameters]
```

Parameters

Table 1024. Parameters

Parameter	Type	Required	Description
pluginId	string	true	The full plugin id of the plugin to be installed.

Related REST command: Install a plug-in.

listArtifacts

Get the list of artifacts for a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  listArtifacts [parameters]
```

Parameters

Table 1025. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID

Table 1025. Parameters (continued)

Parameter	Type	Required	Description
version	string	true	Name or ID of the version
singleFilePath	string	false	Optional path to an individual version artifact to list.

Example

Examples

Note: By default, this command lists only the files and folders in the root folder, as in the following example. The command does not recurse into other folders.

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  listArtifacts
  -component MyComponent
  -version 1.0
```

The result of this command looks like the following example code:

```
[
  {
    "name": "fileA.txt",
    "path": "fileA.txt",
    "version": 1,
    "$ref": "fileA.txt/",
    "type": "file",
    "children": false,
    "lastModified": 1449262065000,
    "length": 0,
    "metaOnly": false
  },
  {
    "name": "folderA",
    "path": "folderA",
    "version": 1,
    "$ref": "folderA/",
    "type": "folder",
    "children": true,
    "lastModified": 1449262065000,
    "length": 0,
    "metaOnly": false
  },
  {
    "name": "folderB",
    "path": "folderB",
    "version": 1,
    "$ref": "folderB/",
    "type": "folder",
```

```
    "children": true,
    "lastModified": 1449262065000,
    "length": 0,
    "metaOnly": false
  }
]
```

Note: To list files and folders that are not in the root folder, you must specify the **singleFilePath** parameter, as in the following example:

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  listArtifacts
  -component MyComponent
  -version 1.0
  -singleFilePath folderA
```

The result of this command includes only the contents of the specified folder, as in the following example:

```
[
  {
    "name": "fileA.txt",
    "path": "folderA/fileA.txt",
    "version": 1,
    "$ref": "folderA/fileA.txt/",
    "type": "file",
    "children": false,
    "lastModified": 1451488971000,
    "length": 0,
    "metaOnly": false
  }
]
```

Related REST command: Get the list of artifacts for a version.

loadEnvironmentCompliancy

Get the compliancy of a given environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  loadEnvironmentCompliancy [parameters]
```

Parameters

Table 1026. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of environment

Table 1026. Parameters (continued)

Parameter	Type	Required	Description
snapshot	string	false	Name or ID of snapshot (required if environment is specified by name)
application	string	false	Name or ID of application (required if snapshot is specified by name)

Related REST command: Get the compliancy of a given environment.

lockSnapshotConfiguration

Lock a snapshot's configuration

This command locks the configuration included in a snapshot. This prevents the snapshot from being affected by any future changes to the relevant configuration and also prevents users from manually changing which configuration the snapshot is using. This cannot be reversed.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  lockSnapshotConfiguration [parameters]
```

Parameters

Table 1027. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Related REST command: Lock a snapshot's configuration.

lockSnapshotVersions

Lock a snapshot's version list

This command locks the list of component versions included in a snapshot. This cannot be reversed and prevents any future changes to the list of versions included in the snapshot.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  lockSnapshotVersions [parameters]
```

Parameters

Table 1028. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID

Related REST command: Lock a snapshot's version list.

lockUser

Lock a user account

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  lockUser [parameters]
```

Parameters

Table 1029. Parameters

Parameter	Type	Required	Description
user	string	true	Name/ID of the user. The name will be case-sensitive if a case-sensitive realm is provided.
realm	string	false	Limit any user name search to given AuthenticationRealm.

Related REST command: Lock a user account.

login

Log in to the server

This command creates an authentication token on the server. Subsequent udclient commands that you run use this token, so you do not have to specify credentials. The token is recorded locally in the user's home directory in the file .ucd/authToken. The token expires 24 hours after it was created. Use the logout command to delete the token and the authtoken file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  login
```

Parameters

None.

Example

Example

```
udclient -username admin -password admin
  -weburl https://myserver.example.com:8443
  login
```

logout

Log out of the server

This command deletes the authentication token that is currently in use, and deletes the local file that contains information about this token.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  logout
```

Parameters

None.

Example

Example

```
udclient
  -weburl https://myserver.example.com:8443
  logout
```

moveResource

Update a resource's parent

To update a resource's parent with this command, you must specify, at a minimum, the resource path or id, and optionally provide info for the resource template, blueprint, and application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  moveResource [parameters] [JSON file]
```

Parameters

Table 1030. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
parent	string	true	The path or id of the new parent

Table 1030. Parameters (continued)

Parameter	Type	Required	Description
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agent": "Name or ID of an agent to use for automation
  (Optional)",
  "agentPool": "Name or ID of an agent pool to use for
  automation (Optional)",
  "component": "Name or ID of a component to associate
  with the resource (Optional)",
  "componentTag": "Name or ID of a component tag to
  associate with the resource; any components with that tag
  can be deployed to this resource (Optional)",
  "description": "Description",
  "impersonationForce": "Specify true to prevent children
  or steps from specifying their own impersonation settings
  (Optional)",
  "impersonationGroup": "Group for default impersonation
  (Optional)",
  "impersonationPassword": "Password for default
  impersonation. Agents running on Linux and Unix machines
  will ignore this value (Optional)",
  "impersonationUseSudo": "Specify true to use the sudo
  command for impersonation. Agents running on Windows
  machines will ignore this value (Optional)",
  "impersonationUser": "User name for default
  impersonation (Optional)",
  "inheritTeam": "Whether or not to set inherit teams for
  resource. teamMappings will be ingored if true. Defaults
  to true (Optional)",
  "name": "Name for the new resource; leave blank to
  inherit the name from an agent, pool, or role",
  "parent": "ID or path to parent resource; leave blank to
  create a root resource (Optional)",
  "role": "Name or ID of a resource role to associate with
  the agent; if this resource is for a component, specify
  the component name instead (Optional)",
  "roleProperties": {"Property name": "Property value
  (Optional)"},
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
```

```
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  moveResource
  -resource /container1/subfolder1a/subfolder1b
  -parent /container2/subfolder2a
```

Related REST command: Update a resource's parent.

pauseWorkflow

Pause a workflow in progress

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  pauseWorkflow [parameters]
```

Parameters

Table 1031. Parameters

Parameter	Type	Required	Description
workflow	string	true	ID of the workflow to pause.

Related REST command: Pause a workflow in progress.

prepareServerKeystoreImportTask

Prepare for importing server keystore entries

Prepares for importing server keystore key pair entries by creating a new import task.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  prepareServerKeystoreImportTask [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "format": "One of 'JKS' or 'PKCS12'.",
  "keystoreEntryPassword": "The password for decrypting
  individual entries. All entries must use the same
  password.",
  "keystorePassword": "The password for decrypting or
  verifying the keystore."
}
```

Related REST command: Prepare for importing server keystore entries.

previewUserLockEvent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  previewUserLockEvent
```

Parameters

None.

Related REST command: Preview user lock event.

provisionEnvironment

Provision a cloud environment

This command provisions an instance of an application blueprint and creates an application environment based on that blueprint.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  provisionEnvironment [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Application name or ID (required if you
  are specifying the blueprint by name",
  "baseResource": "Path to the resource node to create
```

```
  this environment in",
  "blueprint": "Blueprint name or ID",
  "description": "Description (optional)",
  "name": "Environment name",
  "nodeProperties": {"Path to the resource node, such as
  /path/to/resource/node": {"Name of node property": "Value
  or node property"}}
}
```

Example

Examples

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  provisionEnvironment newEnvironment.json
```

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  provisionEnvironment -file newEnvironment.json
```

Example

Example JSON requests

How you structure the JSON request depends on whether you assign the virtual nodes to a cloud group or an environment profile. For more information on cloud groups and environment profiles, see the documentation for your cloud system.

The following example creates an environment that is named `CloudEnv1` from a blueprint that is named `My application`. In this case, the `cloud_group` property is at the root of the JSON string, which specifies that the nodes will receive IP addresses from the same cloud group.

```
{
  "name": "CloudEnv1",
  "application": "My application",
  "baseResource": "/environments",
  "blueprint": "Blueprint 1",
  "description": "New cloud environment from blueprint 1",
  "lockSnapshots": "false",
  "requireApprovals": "false",
  "cloud_group": "2",
  "nodeProperties": {
    "/Deploy Agent/os_part": {
      "multiplicity": "1",
      "numvcpus": "1",
      "memsize": "1024",
      "password": "password",
      "password_0": "password"
    }
  }
}
```

You can also specify IP addresses for the nodes individually. In this case, you specify an environment profile to use for the request and then a cloud group and IP group for each node. The following example has two nodes, and each uses a different cloud group and IP group.

```
{
  "name": "CloudEnv2",
  "application": "My application",
  "baseResource": "/environments",
  "blueprint": "Blueprint 1",
  "description": "New cloud environment from blueprint 1",
  "lockSnapshots": "false",
  "requireApprovals": "false",
  "environment_profile": "1",
  "nodeProperties": {
    "/Deploy Agent/os_part": {
      "cloud_group": "1",
      "ip_group": "1",
      "multiplicity": "1",
      "numvcpus": "1",
      "memsize": "1024",
      "password": "password",
      "password_0": "password"
    },
    "/Deploy Agent/os_part0": {
      "cloud_group": "2",
      "ip_group": "2",
      "multiplicity": "1",
      "numvcpus": "1",
      "memsize": "1024",
      "password": "password",
      "password_0": "password"
    }
  }
}
```

Note: You must specify cloud groups, IP groups, and environment profiles by ID, not by name. In most cases, to get the ID, open the cloud system console, open the group or profile, and find the ID in the URL in your web browser. In other cases, the ID is shown on the cloud system console.

To specify parameters on script packages on the nodes, add the information to the `nodeProperties` array. The following example assumes a script package named `iwd_registration` with a parameter named `cloud_hostname`:

```
{
  "name": "CloudEnv3",
  "application": "My application",
  "baseResource": "/environments",
  "blueprint": "Blueprint 1",
  "description": "New cloud environment from blueprint 1",
  "lockSnapshots": "false",
  "requireApprovals": "false",
  "cloud_group": "2",
  "nodeProperties": {
    "/Deploy Agent/os_part": {
      "multiplicity": "1",
      "numvcpus": "1",
      "memsize": "1024",
      "password": "password",
      "password_0": "password",
      "iwd_registration/cloud_hostname": "cloudserver.example.org"
    }
```

```
  }
}
```

Example

Example response

```
{
  "id": "d84f5683-b05f-4e8b-afdf-f28a38b0fece",
  "securityResourceId": "2eb5db02-b07e-4010-a0cf-3b84509d0ca4",
  "name": "CloudEnv1",
  "description": "New cloud environment",
  "color": "#ffffff",
  "requireApprovals": false,
  "lockSnapshots": false,
  "calendarId": "7aac87db-3984-4cde-98be-a935fd38bdb0",
  "active": true,
  "cleanupDaysToKeep": 0,
  "cleanupCountToKeep": 0,
  "conditions": [
  ]
}
```

Related REST command: Provision a cloud environment.

removeAgentFromPool

Remove an agent from an agent pool

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeAgentFromPool [parameters]
```

Parameters

Table 1032. Parameters

Parameter	Type	Required	Description
pool	string	true	Name or ID of the agent pool
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeAgentFromPool
  -pool Pool1
  -agent Agent1
```

Related REST command: Remove an agent from an agent pool.

removeAgentFromTeam

Remove an agent from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeAgentFromTeam [parameters]
```

Parameters

Table 1033. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Application".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeAgentFromTeam
  -agent MyAgent
  -team MyTeam
```

Related REST command: Remove an agent from a team.

removeAgentPoolFromTeam

Remove an agent pool from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeAgentPoolFromTeam [parameters]
```

Parameters

Table 1034. Parameters

Parameter	Type	Required	Description
agentPool	string	true	Name or ID of the agent pool
team	string	true	Name or ID of the team

Table 1034. Parameters (continued)

Parameter	Type	Required	Description
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard agent pool".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeAgentPoolFromTeam
  -agentPool MyAgentPool
  -team MyTeam
```

Related REST command: Remove an agent pool from a team.

removeAgentServerPin

Remove public key pin for a server host from an agent

This command requests the agent remove a public key pin for a server host, but does not wait for or report the result. The returned maintenance task can be used to query the result.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeAgentServerPin [parameters]
```

Parameters

Table 1035. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
host	string	true	The host name, as it is known by the agent, from which the public key pin is removed
publicKeyFingerprint	string	true	The fingerprint for the public key pin to remove

Related REST command: Remove public key pin for a server host from an agent.

removeApplicationFromTeam

Remove an application from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeApplicationFromTeam [parameters]
```

Parameters

Table 1036. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application.
team	string	true	Name or ID of the team.
type	string	false	Name of the security type to use. Leave blank if you are using the type "Standard Application".

Related REST command: Remove an application from a team.

removeComponentFromApplication

Remove a component from an application.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeComponentFromApplication [parameters]
```

Parameters

Table 1037. Parameters

Parameter	Type	Required	Description
component	string	true	Name of the component. This can be repeated for multiple components.
application	string	true	Name of the application

Related REST command: Remove a component from an application..

removeComponentFromTeam

Remove a component from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeComponentFromTeam [parameters]
```

Parameters

Table 1038. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Component".

Related REST command: Remove a component from a team.

removeComponentResourcePropDef

Delete a component resource property definition

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeComponentResourcePropDef [parameters]
```

Parameters

Table 1039. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the property definition to remove

Related REST command: Delete a component resource property definition.

removeComponentTemplateFromTeam

Remove a component template from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeComponentTemplateFromTeam [parameters]
```

Parameters

Table 1040. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Component Template".

Related REST command: Remove a component template from a team.

removeEnvironmentBaseResource

Remove a base resource from an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeEnvironmentBaseResource [parameters]
```

Parameters

Table 1041. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name of the application; this value is required if you specify the environment name instead of ID
resource	string	true	Path or ID of the resource to remove

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeEnvironmentBaseResource
  -application JPetStore
  -environment "Tutorial environment 1"
  -resource "/JPetStore agents/Tutorial agent 1"
```

Related REST command: Remove a base resource from an environment.

removeEnvironmentFromTeam

Remove an environment from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeEnvironmentFromTeam [parameters]
```

Parameters

Table 1042. Parameters

Parameter	Type	Required	Description
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
environment	string	true	Name or ID of the environment
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Environment".

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeEnvironmentFromTeam
  -application MyApplication
  -environment MyEnvironment
  -team MyTeam
```

Related REST command: Remove an environment from a team.

removeEnvironmentProperty

Delete a component environment property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeEnvironmentProperty [parameters]
```

Parameters

Table 1043. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
name	string	true	Name of the property to remove

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeEnvironmentProperty
  -name MyProperty
  -component d2e59ca4-a2f5-4851-9c85-629c1019ba20
```

Related REST command: Delete a component environment property.

removeGroupFromRoleOnTeam

Remove a group from a role on a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeGroupFromRoleOnTeam [parameters]
```

Parameters

Table 1044. Parameters

Parameter	Type	Required	Description
group	string	true	Name or ID of the group
team	string	true	Name or ID of the team
role	string	true	Name or ID of the role to remove from the group

Related REST command: Remove a group from a role on a team.

removeInventoryForUnmappedComponents

Removes environment inventory for components with no resources in the enviroment resources.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeInventoryForUnmappedComponents [parameters]
```

Parameters

Table 1045. Parameters

Parameter	Type	Required	Description
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID
environment	string	true	Name or ID of the environment

Related REST command: Remove environment inventory for components with no resources in the enviroment resources..

removeProcessFromTeam

Remove a process from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeProcessFromTeam [parameters]
```

Parameters

Table 1046. Parameters

Parameter	Type	Required	Description
process	string	true	Name or ID of the process.
team	string	true	Name or ID of the team.
type	string	false	Name of the security type to use; the default is Standard Process.

Related REST command: Remove a process from a team.

removeResourceFromTeam

Remove a resource from a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeResourceFromTeam [parameters]
```

Parameters

Table 1047. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
team	string	true	Name or ID of the team
type	string	false	Name of the type to use. Leave blank if you are using the type "Standard Resource".

Related REST command: Remove a resource from a team.

removeTagFromAgent

Delete a tag from an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeTagFromAgent [parameters]
```

Parameters

Table 1048. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
tag	string	true	Name of the tag

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeTagFromAgent
  -agent Agent1
  -tag Tag1
```

Related REST command: Delete a tag from an agent.

removeTagFromApplication

Remove a tag from an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeTagFromApplication [parameters]
```

Parameters

Table 1049. Parameters

Parameter	Type	Required	Description
application	string	true	Name or ID of the application
tag	string	true	Name of the tag

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeTagFromApplication
  -application JPetStore
  -tag Tag1
```

Related REST command: Remove a tag from an application.

removeTagFromComponent

Delete a tag from a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeTagFromComponent [parameters]
```

Parameters

Table 1050. Parameters

Parameter	Type	Required	Description
component	string	true	Name or ID of the component
tag	string	true	Name of the tag

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeTagFromComponent
  -component JPetStore-APP
  -tag Tag1
```

Related REST command: Delete a tag from a component.

removeTagFromResource

Remove a tag from a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeTagFromResource [parameters]
```

Parameters

Table 1051. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)
tag	string	true	Name of the tag

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeTagFromResource
  -resource /Resource1/Subresource1
  -tag Tag1
```

Related REST command: Remove a tag from a resource.

removeUserFromGroup

Remove a user from a group

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeUserFromGroup [parameters]
```

Parameters

Table 1052. Parameters

Parameter	Type	Required	Description
user	string	true	Name of the user
group	string	true	Name of the group

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeUserFromGroup
  -user jsmith
  -group group1
```

Related REST command: Remove a user from a group.

removeUserFromRoleOnTeam

Remove a user from a role on a team

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeUserFromRoleOnTeam [parameters]
```

Parameters

Table 1053. Parameters

Parameter	Type	Required	Description
user	string	true	Name or ID of the user
team	string	true	Name or ID of the team
role	string	true	Name or ID of the role to remove from the user

Related REST command: Remove a user from a role on a team.

removeVersionFromSnapshot

Remove a component version from a snapshot

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeVersionFromSnapshot [parameters]
```

Parameters

Table 1054. Parameters

Parameter	Type	Required	Description
snapshot	string	true	Name or ID of the snapshot
application	string	false	Name of the application; this value is required if you specify the snapshot name instead of ID
version	string	true	Name or ID of the component version
component	string	false	Name of the component; this value is required if you specify the component name instead of ID

Related REST command: Remove a component version from a snapshot.

removeVersionLink

Remove a link from a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeVersionLink [parameters]
```

Parameters

Table 1055. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version

Table 1055. Parameters (continued)

Parameter	Type	Required	Description
linkName	string	true	Name of the link to remove

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeVersionLink
  -component JPetStore-APP
  -version 1.0
  -linkName "Link to MyCompany example web site"
```

Related REST command: Remove a link from a version.

removeVersionLinkByName

Remove a link from a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeVersionLinkByName [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Name or ID of the component; this value is
  required if you specify the version name instead of ID",
  "name": "Name of the link",
  "version": "Name or ID of the version"
}
```

Related REST command: Remove a link from a version.

removeVersionStatus

Remove a status from a version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeVersionStatus [parameters]
```

Parameters

Table 1056. Parameters

Parameter	Type	Required	Description
component	string	false	Name or ID of the component; this value is required if you specify the version name instead of ID
version	string	true	Name or ID of the version
status	string	true	Name of the status to apply

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  removeVersionStatus
  -component JPetStore-APP
  -version 1.0
  -status Approved
```

Related REST command: Remove a status from a version.

removeWebhookFromApplication

Removes a webhook from a process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeWebhookFromApplication
```

Parameters

Table 1057. Parameters

Parameter	Type	Required	Description
webhook	string	true	ID of webhook
application	string	true	ID or name of application

Related REST command: Removes a webhook from a process.

removeWebhookFromProcess

Removes a webhook from a process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  removeWebhookFromProcess
```

Parameters

Table 1058. Parameters

Parameter	Type	Required	Description
process	string	true	Id or name of process
webhook	string	true	Id of webhook

Related REST command: Removes a webhook from a process.

repeatApplicationProcessRequest

Repeat an application process request

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  repeatApplicationProcessRequest [parameters]
```

Parameters

Table 1059. Parameters

Parameter	Type	Required	Description
request	string	true	ID of the application process request to repeat

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  repeatApplicationProcessRequest
  -request 47610d75-25c8-420d-a696-f006c34f5423
```

Example

Example response

```
{
  "requestId": "8ece2acb-b8d6-4748-809a-44bf19f4f8f0"
}
```

Related REST command: Repeat an application process request.

requestApplicationProcess

Run an application process

A version may be set to 'latest' to deploy the latest version of that component.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  requestApplicationProcess [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Application name or ID",
  "applicationProcess": "Application process name or ID",
  "date": "Date and time to schedule the process for.
  (Optional) Supports unix timecodes or the format yyyy-mm-
  dd HH:mm",
  "description": "Description for the request (Optional)",
  "environment": "Environment name or ID",
  "onlyChanged": "Specify false to force deployment of
  versions that are already in the inventory (Optional)",
  "post-deploy-message": "The body of the PUT message. You
  can use the variable ${p:finalStatus}, which holds the
  state of the process. The possible states are: Success,
  Failure, Approval Rejected, Awaiting Approval, Running,
  Scheduled, Cancelled, and Unknown. You can use the
  variable ${p:warningCount}, which holds the number of
  warnings that occurred on the process. (Optional)",
  "post-deploy-put-url": "The URL that the post-deploy-
  message is PUT to. (Optional)",
  "properties": {"Property name": "Property value
  (Optional)"},
  "recurrencePattern": "To make a scheduled process recur,
  specify 'D' (daily), 'W' (weekly), 'M' (monthly), or 'WD'
  (week days). (Optional)",
  "snapshot": "Snapshot name or ID (Optional)",
  "versions": [{
    "component": "Component name or ID for the version, if
  you are using version name instead of ID.",
    "version": "Version name or ID (Repeat as necessary.
  Not used with snapshots)"
```

```
  }]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  requestApplicationProcess runProcess.json
```

Example

Example JSON request

```
{
  "application": "JPetStore",
  "description": "Requesting deployment",
  "applicationProcess": "Deploy JPetStore",
  "environment": "Tutorial environment 1",
  "onlyChanged": false,
  "properties": {
    "Prop1": "value1"
  },
  "versions": [
    {
      "version": "1.0",
      "component": "JPetStore-APP"
    },
    {
      "version": "1.0",
      "component": "JPetStore-WEB"
    },
    {
      "version": "1.0",
      "component": "JPetStore-DB"
    }
  ]
}
```

Example

Example JSON request

You can also use the keywords `latest`, `latestVersion`, and `newest` in place of component version names. In this case, the process uses the most recently created component version. The following example shows how you can use these keywords in the JSON request.

```
{
  "application": "JPetStore",
  "description": "Deploying newest versions",
  "applicationProcess": "Deploy JPetStore",
  "environment": "Tutorial environment 1",
  "onlyChanged": false,
  "properties": {
    "Prop1": "value1"
  },
```

```
  "versions": [
    {
      "version": "newest",
      "component": "JPetStore-APP"
    },
    {
      "version": "latest",
      "component": "JPetStore-WEB"
    },
    {
      "version": "latestVersion",
      "component": "JPetStore-DB"
    }
  ]
}
```

Example

Example response

```
{
  "requestId": "f7e7b00d-8ea6-4a95-ad74-0ff853125232"
}
```

Related REST command: Run an application process.

requestProcess

Executes new generic process request(s) for resource(s)

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  requestProcess [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "processId": "ID of generic process",
  "processVersion": "(Optional) Version number of the
  process to run against. -1 (default) represents latest
  version",
  "properties": {"Property name": "Property value
  (Optional)"},
  "resource": "Deprecated, use 'resources'(ID or path of
  resource to run the process against)",
  "resources": ["Comma-separated list of IDs or paths of
  resources to run the process against. 'resource' parameter
  would be ignored if this is present"]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  requestProcess [JSON File]
```

Example

Example response

```
[{
  "id": "1710e3a8-b52f-02e3-d5ed-421c17f24af2",
  "submittedTime": 1585081649980,
  "userName": "admin",
  "processPath": "processes\/1710e0b0-eb7e-e9b7-38ab-c7be3e828f56",
  "processVersion": 4
}]
```

Example

Example response

```
[{
  "id": "1710e506-b23a-ab88-7634-563464e20ef0",
  "submittedTime": 1585083083529,
  "userName": "admin",
  "processPath": "processes\/1710e0b0-eb7e-e9b7-38ab-c7be3e828f56",
  "processVersion": 2
}]
```

Example

Example response

```
[
  {
    "id": "1710e526-9d32-1fc1-299f-f7562b3b33f3",
    "submittedTime": 1585083214272,
    "userName": "admin",
    "processPath": "processes\/1710e0b0-eb7e-e9b7-38ab-c7be3e828f56",
    "processVersion": 4
  },
  {
    "id": "1710e526-9ee8-a8f1-676c-dbb25d9885a2",
    "submittedTime": 1585083214274,
    "userName": "admin",
    "processPath": "processes\/1710e0b0-eb7e-e9b7-38ab-c7be3e828f56",
    "processVersion": 4
  }
]
```

Example

Example response

```
[{
  "id": "1710e560-809e-477f-d5ef-d260dbf4578c",
  "submittedTime": 1585083451381,
  "userName": "admin",
  "processPath": "processes\/1710e0b0-eb7e-e9b7-38ab-c7be3e828f56",
  "processVersion": 4
}]
```

Related REST command: Executes new generic process request(s) for resource(s).

restartAgent

Restart an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  restartAgent [parameters]
```

Parameters

Table 1060. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  restartAgent
  -agent Agent1
```

Related REST command: Restart an agent.

restoreArtifacts

Restore artifacts of an archived version

Restore Artifacts of an archived version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  restoreArtifacts [parameters]
```

Parameters

Table 1061. Parameters

Parameter	Type	Required	Description
component	string	false	Name/ID of the Component (Optional, required if version name specified instead of id
version	string	true	Name/ID of the version

Related REST command: Restore artifacts of an archived version.

restoreFromRecycleBin

Restore from recycle bin.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  restoreFromRecycleBin [parameters]
```

Parameters

Table 1062. Parameters

Parameter	Type	Required	Description
resource	string	true	Id of the resource to be restored. This parameter can be repeated to restore multiple resources at once.

Related REST command: Restore from recycle bin..

resumeWorkflow

Resume a workflow in progress

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  resumeWorkflow [parameters]
```

Parameters

Table 1063. Parameters

Parameter	Type	Required	Description
workflow	string	true	ID of the workflow to resume.

Related REST command: Resume a workflow in progress.

revokeAuthToken

Revoke an authentication token

Revokes and deletes an authentication token identified by database ID or token secret. Only users with 'Manage Security' permission can use this function for a token they do not own.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  revokeAuthToken [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"token": "Authentication token ID or secret"}
```

Example

Example command to delete auth tokens using a JSON file

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  revokeAuthToken filename.json
```

Example

JSON example

```
{
  "token": "194dba21-0199-cddb-1f04-4ad922fc5385"
}
```

Related REST command: Revoke an authentication token.

rollbackZInventoryEntries

Rollback Z inventory.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  rollbackZInventoryEntries [parameters]
```

Parameters

Table 1064. Parameters

Parameter	Type	Required	Description
application	string	false	Application name or id. This is required when specifying environment by name.
environment	string	true	Environment name or id.
component	string	false	Component name or id. This is required when specifying version by name.
version	string	true	Version name or id.
overwriteUpdatedFiles	boolean	false	Whether or not to rollback files updated by later versions. Defaults to true for backwards compatability.

Related REST command: Rollback Z inventory..

saveEnvironmentGate

Save the environment gate of a given environment. Environment gates are JSON objects consisting of operators "OR", "AND", and "NOT", as well as status UUIDs. A valid operator is a JSON Object with a single key corresponding to one of the three operator types, and with a list of operators or UUIDs as its value. A valid gate begins with an enclosing OR operator containing one or more AND operators. AND operators can contain only UUIDs or NOT operators. NOT operators must contain a single UUID which corresponds to a known version or snapshot status.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  saveEnvironmentGate [parameters] [JSON file]
```

Parameters

Table 1065. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of environment

Table 1065. Parameters (continued)

Parameter	Type	Required	Description
application	string	false	Name or ID of application (required if environment is specified by name)

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"OR": [{"AND": [
  "<uuid>",
  "<uuid>",
  "...",
  {"NOT": ["<uuid>"]},
  {"NOT": ["<uuid>"]},
  "..."
]}]}
```

Related REST command: Save the environment gate of a given environment.

saveEnvironmentTemplateGate

Save the environment gate of a given environment template. Environment gates are JSON objects consisting of operators "OR", "AND", and "NOT", as well as status UUIDs. A valid operator is a JSON Object with a single key corresponding to one of the three operator types, and with a list of operators or UUIDs as its value. A valid gate begins with an enclosing OR operator containing one or more AND operators. AND operators can contain only UUIDs or NOT operators. NOT operators must contain a single UUID which corresponds to a known version or snapshot status.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  saveEnvironmentTemplateGate [parameters] [JSON file]
```

Parameters

Table 1066. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	Name or ID of environment template
applicationTemplate	string	false	Name or ID of applicationTemplate (required if environment template is specified by name)

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"OR": [{"AND": [
  "<uuid>",
  "<uuid>",
  "...",
  {"NOT": ["<uuid>"]},
  {"NOT": ["<uuid>"]},
  "..."
]}]}
```

Related REST command: Save the environment gate of a given environment template.

savePostProcessingScript

Get information about a post processing script

This command returns a JSON representation of a post processing script.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  savePostProcessingScript [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "body": "The body of the script (required)",
  "description": "The description of the script (optional)",
  "id": "The id of the script (required)",
  "name": "The name of the script (required)",
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ]
}
```

Related REST command: Get information about a post processing script.

searchZInventoryEntries

Search Z inventory.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  searchZInventoryEntries [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applications": ["Application name or id. Only one of
  applications, environments, or components may be
  specified."],
  "components": ["Component name or id. Only one of
  applications, environments, or components may be
  specified."],
  "environments": ["Environment name or id. Only one of
  applications, environments, or components may be
  specified."],
  "fileNameGlob": "File names to search for. Either this
  or versionNameGlob must be specified. Only one of
  fileNameGlob or versionNameGlob may be specified when
  filtering by applications, environments or components.",
  "includeInactive": "Boolean: default false. Whether or
  not to include inactive entries in the results.",
  "versionNameGlob": "Version names to search for. Either
  this or fileNameGlob must be specified. Only one of
  fileNameGlob or versionNameGlob may be specified when
  filtering by applications, environments or components."
}
```

Related REST command: Search Z inventory..

setAgentJavaHome

Set the configured JAVA_HOME of an agent

This command requests the agent set JAVA_HOME, but does not wait for or report the result. The new JAVA_HOME will not take effect until the agent is restarted. The returned maintenance task can be used to query the result.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setAgentJavaHome [parameters]
```

Parameters

Table 1067. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
javahome	string	true	Name of managed Java install or absolute path of a valid JAVA_HOME directory

Related REST command: Set the configured JAVA_HOME of an agent.

setAgentKeypairPolicy

Set or modify an agent's keypair policy

This command requests the agent set or modify its keypair policy, but does not wait for or report the result. The returned maintenance task can be used to query the result. If the maximum age of the keypair is changed to be less than its current age, keypair rotation may begin immediately, but the outcome of the returned maintenance task does not depend on that.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setAgentKeypairPolicy [parameters]
```

Parameters

Table 1068. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
maxAgeDays	int	true	The maximum age in days of the agent's keypair, or zero to disable rotation

Related REST command: Set or modify an agent's keypair policy.

setAgentLicenseType

Set the license for an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setAgentLicenseType [parameters]
```

Parameters

Table 1069. Parameters

Parameter	Type	Required	Description
licenseType	string	true	The License Type to set on the agent(s)
agent	string	true	Name or ID of the agent. This can be repeated for multiple agents.

Related REST command: Set the license for an agent.

setAgentProperty

Set a property on a agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setAgentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agent": "Name or ID of the agent",
  "description": "The description of the property
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property",
  "value": "New value for the property (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setAgentProperty
  -agent Agent1
  -name Prop1
  -value value1
```

Example

Example response

```
{
  "id":"cdf5e901-f19a-48b8-82fc-c92c256e31df",
  "name":"newProperty",
  "value":"newValue",
  "secure":false
}
```

Related REST command: Set a property on a agent.

setAgentUnavailable

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setAgentUnavailable [parameters]
```

Parameters

Table 1070. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
unavailable	boolean	true	The desired unavailable state.

Related REST command: Set an agent unavailable.

setApplicationProperty

Set a property on an application

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setApplicationProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or ID of the application",
  "description": "The description of the property
```

```
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Property name",
  "value": "Property value (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setApplicationProperty
  -application JPetStore
  -name Prop4
  -value value4
```

Example

Example response

```
{
  "id": "7b818dfe-efd9-465a-bee3-775f60dc297d",
  "name": "Prop4",
  "value": "value4",
  "secure": false
}
```

Related REST command: Set a property on an application.

setComponentEnvironmentProperty

Set a component environment property value

This command will set a component property value specifically for this environment.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setComponentEnvironmentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or ID of the application; this
  value is required if you specify the environment name
  instead of ID",
  "component": "Name or ID of the component",
```

```
  "environment": "Name or ID of the environment",
  "name": "Name of the property to set",
  "value": "Value of the property to set (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setComponentEnvironmentProperty
  -name Prop1
  -value value1
  -component JPetStore-APP
  -environment Environment1
  -application JPetStore
```

Related REST command: Set a component environment property value.

setComponentProperty

Set a property on a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setComponentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Name or ID of the component",
  "description": "The description of the property
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property",
  "value": "Value of the property (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setComponentProperty
  -component JPetStore-APP
```

```
  -name Prop1
  -value value1
```

Example

Example response

```
{
  "id": "3702f391-09e1-4550-903a-d5ae286c94aa",
  "name": "Prop1",
  "value": "value1",
  "secure": false
}
```

Related REST command: Set a property on a component.

setComponentVersionPropDef

Set a version property for a component

Component version properties are available to each version of the component.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setComponentVersionPropDef [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "allowedValues": "Comma separated list of allowed values
  in SELECT or MULTI_SELECT properties (Optional)",
  "component": "Name or ID of the component",
  "description": "Description for the property (Optional)",
  "label": "Label of the property (Optional)",
  "name": "Name of the property to set",
  "pattern": "A regular expression that specifies valid
  values for the property. Follows java.util.regex.Pattern
  (Optional)",
  "required": "Whether the property is required (Optional,
  Boolean)",
  "type": "the type of property definition; valid values
  are TEXT (the default), , TEXTAREA, CHECKBOX, SELECT,
  MULTI_SELECT, DATETIME, and SECURE)",
  "value": "Default value of the property definition
  (Optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setComponentVersionPropDef
  -component JPetStore-APP
  -name Prop1
  -value true
  -type CHECKBOX
```

Example

Example response

```
{
  "id": "870bc2f5-55e2-4cb9-b3ab-f750688754b9",
  "name": "Prop1",
  "type": "CHECKBOX",
  "value": "prop1",
  "required": false
}
```

Related REST command: Set a version property for a component.

setEnvironmentProperty

Set a property on environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setEnvironmentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or ID of the application; this
  value is required if you specify the environment name
  instead of ID",
  "description": "The description of the property
  (optional)",
  "environment": "Name or ID of the environment",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property to set",
  "value": "Value of the property to set (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setEnvironmentProperty
  -application JPetStore
  -environment "Tutorial environment 1"
  -name Prop1
  -value value1
```

Example

Example response

```
{
  "id": "ba3f8bd4-c26e-4529-a2a3-66e283242244",
  "name": "Prop1",
  "value": "value1",
  "secure": false
}
```

Related REST command: Set a property on environment.

setLoggingConfig

Get logging configuration

Sets the current logging configuration. The output of the getLoggingConfig command is acceptable input. To support this, several elements of the input template are optional and ignored, but must be valid if supplied. Regardless, the input is still subject to the constraints reported by these ignored elements. For example, objects for which 'deletable' is false must be present in the input as their absense is interpreted as a request for deletion.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setLoggingConfig [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "createdDate": "Timestamp when the object was created;
  ignored and optional",
  "logs": {
    "createdDate": "Timestamp when the object was created;
  ignored and optional",
    "deletable": "Reports if log can be deleted; ignored
  and optional",
    "enabled": "Reports if log is enabled; the main log
  cannot be disabled",
    "enabledMutable": "Reports if 'enabled' can be
```

```
  changed; ignored and optional",
    "fileCount": "Number of rotated logs; must be at least
  1",
    "fileMegabytes": "Maximum log file size; ignored and
  optional if rotation is not SIZE",
    "loggers": [{
      "createdDate": "Timestamp when the object was
  created; ignored and optional",
      "deletable": "Reports if the logger can be deleted;
  ignored and optional",
      "level": "The logging level; one of
  OFF|FATAL|ERROR|WARN|INFO|DEBUG|TRACE|ALL",
      "logMutable": "Reports if the log assoctation can be
  changed; ignored and optional",
      "modifiedDate": "Timestamp when the object last
  modified; ignored and optional",
      "name": "Name of the logger; must be unique across
  all logs; '*' is the name of the root logger which must
  only be in the main log; up to 255 printable ASCII
  characters (no whitespace or control codes)",
      "root": "Reports if this logger is the root logger;
  ignored and optional"
    }],
    "main": "Reports if this log is the main log; ignored
  and optional",
    "modifiedDate": "Timestamp when the object last
  modified; ignored and optional",
    "name": "Name of the log; must be unique; up to 64
  characters from [-_0-9a-z]",
    "rotation": "Rotation strategy: one of
  SIZE|HOURLY|DAILY|WEEKLY|MONTHLY",
    "rotationMutable": "Reports if 'rotation' can be
  changed; ignored and optional"
  },
  "modifiedDate": "Timestamp when the object last
  modified; ignored and optional"
}
```

Related REST command: Get logging configuration.

setPreferredJavaPackage

Sets a package as preferred for its OS and CPU, unsetting the previously preferred package if there was one.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setPreferredJavaPackage [parameters]
```

Parameters

Table 1071. Parameters

Parameter	Type	Required	Description
package	string	true	Name or ID of the java package

Related REST command: Set a package as preferred for its OS and CPU.

setResourceProperty

Set a property on a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setResourceProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or ID of the application (Required
  to look up blueprint by name)",
  "blueprint": "Name or ID of the blueprint (optional)",
  "description": "The description of the property
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property",
  "resource": "ID of resource or Path to the resource,
  such as /ResourceGroup/Agents/Agent1",
  "template": "Name or ID of the resource template
  (optional)",
  "value": "New value for the property (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setResourceProperty
  -resource "/My resource"
```

```
  -name property1
  -value value1
```

Example

Example response

```
{
  "id": "bd20924e-8547-4d41-b9b1-36b41278d224",
  "name": "property1",
  "value": "value1",
  "secure": false
}
```

Related REST command: Set a property on a resource.

setResourceRoleProperty

Set a property on a resource

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setResourceRoleProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "application": "Name or ID of the application (Required
  to look up blueprint by name)",
  "blueprint": "Name or ID of the blueprint (optional)",
  "description": "The description of the property
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property",
  "resource": "ID of resource or Path to the resource,
  such as /ResourceGroup/Agents/Agent1",
  "template": "Name or ID of the resource template
  (optional)",
  "value": "New value for the property (optional)"
}
```

Related REST command: Set a property on a resource.

setSystemConfiguration

Update the system configuration

Use this command to set any of the properties on the System Settings page of the server. The JSON template for this command lists all of the properties that you can change, but you can omit properties that you are not changing.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setSystemConfiguration [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agentAutoLicense": "Specify true to automatically
  assign licenses to agents",
  "artifactAgent": "Specify a default agent for component
  version imports.",
  "auditLogCleanupEnabled": "Whether the audit logs will
  be deleted after a given number of days.",
  "auditLogCleanupHour": "The hour when audit logs are
  cleaned; the value must be an integer from 0 (midnight) to
  23 (11pm).",
  "auditLogCleanupMinute": "The minute when audit logs are
  cleaned; the value must be an integer from 0 to 59.",
  "auditLogReadEntriesEnabled": "Controls whether READ
  events are captured in the audit log.",
  "auditLogRetentionLength": "The number of days to retain
  audit log entries.",
  "bearerTokenRealm": "The realm to use for bearer
  authentication.",
  "cleanupArchivePath": "The path for archived component
  versions",
  "cleanupCountToKeep": "The number of component versions
  to keep",
  "cleanupDaysToKeep": "The number of days component
  versions are kept",
  "cleanupHourOfDay": "The time when versions are cleaned;
  the value must be an integer from 0 (midnight) to 23 (11
  pm)",
  "codestationMaxSizeDisplayMBs": "Defines the size in
  megabytes of '100%' storage in the dashboard server
  storage display.",
  "defaultLocale": "The default locale to use for the
  system (determines language and date formats.)",
  "deleteEnvResources": "Enable deleting resources
  globally from an environment's resource tree or when
  deleting an environment.",
  "deployMailHost": "The host name of the mail server that
  is used for notifications",
  "deployMailPassword": "The user password for sending
  email notifications",
  "deployMailPort": "The SMTP port that is used by the
  notifications mail server",
  "deployMailSecure": "Specifies whether the SMTP
```

```
connection is secure",
"deployMailSender": "The sender address for email
notifications",
"deployMailUsername": "The user name for sending email
notifications",
"enableAllowFailure": "Controls whether the 'allow
failure' option can be used on the steps in a workflow.",
"enableInactiveLinks": "Specify true to show links to
inactive objects",
"enablePromptOnUse": "Controls whether user-entered
values can be specified at process run time",
"enableUIDebugging": "Changes how the UI resources are
loaded so it is easier to debug problems in the UI if set.
Enabling reduces UI performance.",
"envCompPropsOverrideEnvProps": "Determines whether or
not environment component properties will override
environment properties.",
"environmentCountToKeep": "The number of versions to
retain for this environment. Lists the versions in order
of the time of deployment to this environment, and exempts
the first N from artifact cleanup. If an archive path is
set, the versions that are cleaned up will be archived. If
no archive path is set, the versions will be deleted. To
retain all versions of a component, enter -1.",
"environmentDaysToKeep": "The number of days since
deployment to retain component versions which have been
deployed to this environment. If an archive path is set,
the versions that are cleaned up will be archived. If no
archive path is set, the versions will be deleted. To
retain component versions indefinitely, enter -1.",
"externalApprovalAgent": "Specify a default agent to use
for running External Approval Processes. Superseded by
approvalAgentPool.",
"externalApprovalAgentPool": "Specify a default agent
pool to use for running External Approval Processes.
Supersedes approvalAgent.",
"externalURL": "The URL that agents use to connect to
the server",
"externalUserURL": "The URL that users enter to connect
to the server",
"failProcessesWithUnresolvedProperties": "Determines
whether or not a process will fail when it encounters a
property that cannot be resolved.",
"historyCleanupDaysToKeep": "The number of days
application process history is to be kept. -1 will disable
application process history cleanup.",
"historyCleanupDuration": "The maxiumum number of hours
per day the history cleanup process can take; the value
must be an integer from 1 to 23.",
"historyCleanupEnabled": "The system will not delete any
process history unless this is enabled.",
"historyCleanupHour": "The time when application process
history is cleaned; the value must be an integer from 0
(midnight) to 23 (11pm).",
"historyCleanupMinute": "The minute when process history
is cleaned; the value must be an integer from 0 to 59.",
"integrationTag": "Specify an agent tag to use any agent
with the given tag for version imports.",
```

```
"licenseServerUrl": "The host name and port of the
license server",
"llmApiKey": "The API key used to make requests to the
AI provider.",
"llmEndpoint": "The base endpoint used to construct
requests to the AI provider.",
"llmModel": "The Model name or id. Only modifiable if
llmVendor, llmApiKey, and llmEndpoint are present.",
"llmOrganization": "The organization ID for the relevant
organization.",
"llmProject": "The project ID for the relevant project.",
"llmVendor": "The AI provider used to service AI
requests.",
"maintenanceModeEnabled": "When this option is enabled,
the server will be in maintenance mode.",
"messageOfTheDay": "Message you want to display",
"minimumPasswordLength": "Determines the minimum length
for passwords",
"repoAutoIntegrationPeriod": "The number of seconds
between the times that the server polls components for new
versions",
"requireCommentForProcessChanges": "When enabled, the
user must provide a comment when saving changes to any
process design.",
"requireDigits": "Require new passwords to contain at
least one number character.",
"requireDuplicate": "Require new passwords to not match
the username, actual name, or configured email.",
"requireLowercase": "Require new passwords to contain at
least one lowercase character.",
"requireSpecials": "Require new passwords to contain at
least one special character.",
"requireUppercase": "Require new passwords to contain at
least one uppercase character.",
"showCompliancyOnLoad": "Load visible environment
compliancies when viewing the application page, rather
than on demand.",
"snapshotDaysToKeep": "The number of days to retain
snapshots. Snapshots are always kept if they are the most
recently deployed snapshot in an environment. Each
environment  can override this setting and retain
snapshots longer. In this case a snapshot will only be
deleted if it is not retained by an environment. If no
environment is retaining the snapshot and the snapshot is
older than the number of days in this setting, the server
deletes the snapshot. To retain all snapshots
indefinitely, enter -1.",
"use.default.atr.if.not.specified": "Use the Default
Auth Token Restriction for plugin steps that do not have
an auth token restriction set. If false this will not
restrict auth token usage at all. Any new version will
have an auth token restriction.",
"userIdleDaysToDelete": "The number of days before
deleting a user account due to inactivity.",
"userIdleDaysToLock": "The number of days before locking
a user account due to inactivity.",
"validateAgentIp": "Controls validation of agent IPs and
hostnames. The first time the agent connects, its IP and
```

```
  hostname will be saved. In subsequent connections, if
  either has changed, the agent will be prevented from
  coming online."
}
```

Example

Example JSON request

```
{
  "minimumPasswordLength": "9",
  "requireComplexPasswords": "true"
}
```

Related REST command: Update the system configuration.

setSystemProperty

Set a system property

System properties behave like global variables for the server.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setSystemProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "The description of the property
  (optional)",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "label": "The label of the property (optional)",
  "name": "Name of the property",
  "value": "Value of the property (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setSystemProperty
  -name "System property 1"
  -value value001
```

Example

Example response

```
{
  "id": "f3bec2ee-3f3e-461f-b5b0-e844efafbdbe",
  "name": "System property 1",
  "value": "value001",
  "secure": false
}
```

Related REST command: Set a system property.

setVersionProperty

Set a property on a component version

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  setVersionProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Name or ID of the component; this value is
  required if you specify the version name instead of ID",
  "isSecure": "Whether the property is secure; the default
  is the current state of the property, or false if the
  property is not yet specified (optional)",
  "name": "Name of the property to set",
  "value": "Value of the property to set (optional)",
  "version": "Name or ID of the version"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  setVersionProperty
  -component JPetStore-APP
  -version 1.0
  -name Prop1
  -value value1
```

Example

Example response

```
{
  "id": "33528ab5-a7be-46cb-9f0c-916a42bd2f62",
```

```
  "name": "Prop1",
  "value": "value1",
  "secure": false
}
```

Related REST command: Set a property on a component version.

shutdownAgent

Shut down an agent

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  shutdownAgent [parameters]
```

Parameters

Table 1072. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  shutdownAgent
  -agent Agent1
```

Related REST command: Shut down an agent.

testAgent

Test the connection from an agent to the server

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  testAgent [parameters]
```

Parameters

Table 1073. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  testAgent
  -agent Agent1
```

Related REST command: Test the connection from an agent to the server.

testVaultAppRole

Test resolving a value from a Vault AppRole.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  testVaultAppRole [parameters] [JSON file]
```

Parameters

Table 1074. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultAppRole	string	true	Name or ID of the Vault App Role

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "property": "Property to resolve from vault. i.e.
  vault/KV/Holder:key. (Required)",
  "secretId": "The secretId to use for authentication.
  (Required if not set on Vault AppRole. Ignored if already
  set on Vault AppRole)"
}
```

Related REST command: Test resolving a value from a Vault AppRole..

testVaultCertificate

Test resolving a value from a Vault Certificate.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  testVaultCertificate [parameters] [JSON file]
```

Parameters

Table 1075. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultCertificate	string	true	Name or ID of the Vault App Role

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"property": "Property to resolve from vault. i.e.
  vault/KV/Holder:key. (Required)"}
```

Related REST command: Test resolving a value from a Vault Certificate..

testVaultLdapUser

Test resolving a value from a Vault Ldap User.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  testVaultLdapUser [parameters] [JSON file]
```

Parameters

Table 1076. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultLdapUser	string	true	Name or ID of the Vault Ldap User

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "password": "The Vault Ldap User Password (Required if
  not set on ldap user. Ignored if already set on ldap
  user)",
  "property": "Property to resolve from vault. i.e.
  vault/KV/Holder:key. (Required)"
}
```

Related REST command: Test resolving a value from a Vault Ldap User..

uninstallAgentJava

Uninstall a managed Java installation

This command requests the agent uninstall Java, but does not wait for or report the result. If an installation is the active JAVA_HOME or configured as JAVA_HOME, that installation is not removed.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  uninstallAgentJava [parameters]
```

Parameters

Table 1077. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent
package	string	true	Name of a package to uninstall from the agent

Related REST command: Uninstall a managed Java installation.

unlockUser

Unlock a user account and clear failed login attempts.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  unlockUser [parameters]
```

Parameters

Table 1078. Parameters

Parameter	Type	Required	Description
user	string	true	Name/ID of the user. The name will be case-sensitive if a case-sensitive realm is provided.
realm	string	false	Limit any user name search to given AuthenticationRealm.

Related REST command: Unlock a user account and clear failed login attempts..

unsetPreferredJavaPackage

Unsets a package as preferred for its OS and CPU, leaving no package as preferred.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  unsetPreferredJavaPackage [parameters]
```

Parameters

Table 1079. Parameters

Parameter	Type	Required	Description
package	string	true	Name or ID of the java package

Related REST command: Unset a package as preferred for its OS and CPU.

updateAgent

Update agent settings

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateAgent [parameters] [JSON file]
```

Parameters

Table 1080. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "configurationTemplate": "Name or id of the Agent
  Configuration Template to set for this agent",
  "description": "The description of the agent",
  "impersonationForce": "Force impersonation. When set to
  true, do not allow steps or resource configuration to
  change impersonation settings",
  "impersonationGroup": "Default group to use for
  impersonation",
  "impersonationPassword": "Password to use for
  impersonation (only applicable for Windows)",
```

```
  "impersonationUseSudo": "Use sudo for impersonation
  (only applicable for *nix)",
  "impersonationUser": "Default user to use for
  impersonation",
  "licenseType": "License type to use for this agent",
  "name": "Name of the agent"
}
```

Related REST command: Update agent settings.

updateApplicationPipeline

Update an application pipeline definition

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateApplicationPipeline [parameters] [JSON file]
```

Parameters

Table 1081. Parameters

Parameter	Type	Required	Description
application	string	false	Application name or id. Required if pipeline specified by name.
pipeline	string	true	Pipeline name or id.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "name of the application pipeline",
  "stages": [{
    "blocks": [{
      "environment": "name of the environment for this
  block",
      "processes": ["process-name"]
    }],
    "name": "name of the stage"
  }]
}
```

Related REST command: Update an application pipeline definition.

updateApplicationTemplate

Update an application template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateApplicationTemplate [parameters] [JSON file]
```

Parameters

Table 1082. Parameters

Parameter	Type	Required	Description
applicationTemplate	string	true	The name or ID of an existing application template

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Application template description
  (Optional)",
  "enforceCompleteSnapshots": "Specify true to require an
  explicit version for each component",
  "name": "Application template name",
  "notificationScheme": "Notification scheme (Optional)",
  "onlyChangedVersions": "If true, the 'Only Deploy
  Changed Versions' deployment option will be checked by
  default (default: true)",
  "tagRequirements": [
    {
      "name": "Name of the Component tag",
      "number": "Number of components for the tag",
      "type": "GREATER_THAN, LESS_THAN, or EQUALS"
    },
    " (Optional)"
  ],
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    " (Optional)"
  ],
  "useWizard": "Specify true to use the Deployment Wizard
  when submitting application process request (default:
  false)"
```

```
}
```

Related REST command: Update an application template.

updateApplicationTemplatePropDefs

Update an application template prop sheet def from a JSON file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateApplicationTemplatePropDefs [parameters] [JSON file]
```

Parameters

Table 1083. Parameters

Parameter	Type	Required	Description
applicationTemplate	string	true	Name or ID of the application template

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "allowedValues": "Optional, array of elements {\"
  value\":\"Required, a value\",\"label\":\"Optional,
  defaults to value\"}",
    "description": "Optional",
    "httpAuthenticationType": "Optional, BASIC or BEARER",
    "httpBasePath": "Optional",
    "httpFormat": "JSON or XML",
    "httpLabelPath": "Optional",
    "httpPassword": "Optional",
    "httpUrl": "Optional, required for Http property types",
    "httpUsername": "Optional",
    "httpValuePath": "Optional",
    "label": "Optional, label displayed to user",
    "name": "Required, the property name",
    "pattern": "Optional, regular expression for the value",
    "required": "Optional boolean, if the property is
  required",
    "resolveHttpValuesUrl": "Optional",
    "type": "Optional, Default TEXT, Type of property
  definition",
    "value": "Optional, Default value"
  }]
}
```

Related REST command: Update an application template prop sheet def from a JSON file..

updateAuthenticationRealm

Update the basic settings of an AuthenticationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateAuthenticationRealm [JSON file]
```

Parameters

Table 1084. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the AuthenticationRealm

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "authorizationRealmIds": ["AuthorizationRealms"],
  "description": "A descrption (optional)",
  "name": "AuthenticationRealm name",
  "properties": {
    "auth-endpoint": "OIDC: The URI for authorization with
  the OpenID Provider",
    "authentication-mech": "LDAP: Simple or DIGEST-MD5",
    "client-id": "OIDC: The Client ID which is registered
  with the OpenID Provider",
    "client-secret": "OIDC: The Client Secret which is
  provided by the OpenID Provider upon registration",
    "connection-name": "LDAP: The username to use for
  Simple binding to LDAP for searches. If not specified, an
  anonymous connection is made.",
    "connection-password": "LDAP: The password to use for
  Simple binding to LDAP for searches.",
    "email-claim": "OIDC: The claim containing the user's
  email.",
    "end-session-endpoint": "OIDC: Optional: The URI for
  RP-initiated logout",
    "end-session-id-token-hint": "OIDC: If an ID token
  hint parameter (id_token_hint) is appended to the End
  Session Endpoint URL.",
    "groups-delim": "SSO: A string that acts as the
  delimiter between groups that are passed in the headers.
  For example, if the user should be added to groups GROUP1
  and GROUP2, and the header value was \"GROUP=GROUP1;
  GROUP=GROUP2\" the delimiter would be ';'. Note: special
  regex characters such as '+' must be escaped with one
  backslash ''.",
```

```
    "groups-header": "SSO: The header name that denotes
  the user groups to which user will be added.",
    "groups-regex": "SSO: The regular expression to find
  the groups in the header value. If a capturing group is
  specified, only the first will be caught (per delimited
  string). If no capturing group is specified with
  parentheses, then the whole delimited string will be
  captured.",
    "issuer": "OIDC: The issuer identifier for the issuer
  of the token response.",
    "jwks-uri": "OIDC: The JSON Web Key Sets URI endpoint",
    "name-claim": "OIDC: The claim containing user's name.",
    "sasl-realm": "LDAP: The sasl Realm for DIGEST-MD5
  authentication",
    "token-endpoint": "OIDC: 'The URI for token requests
  with the OpenID Provider",
    "url": "LDAP: The URL of the LDAP server. It should
  begin with 'ldap://' or 'ldaps://'. Failover servers can
  be added by separating the URLs with a space. Example:
  ldap://ldap.mydomain.com:389 ldap://ldap.mydomain2.com",
    "user-base": "LDAP: When you search multiple
  directories, specify the starting directory that is used
  for searches, such as ou=employees,dc=mydomain,dc=com.",
    "user-pattern": "LDAP: LDAP users exist in a single
  directory; use a pattern to create the DN for users.",
    "user-search": "LDAP: The LDAP filter expression to
  use when searching for user directory entries. The
  username is put in place of {0} in the search pattern. If
  this is an attribute and not part of the user DN, wrap in
  parentheses. E.g. uid={0} or (sAMAccountName={0})",
    "user-search-subtree": "LDAP: Search the full subtree
  for the user, as opposed to a single-level search only
  covering users directly inside the specified search base.
  \"true\" or \"false\"",
    "userinfo-endpoint": "OIDC: The URI for userinfo
  requests with the OpenID Provider"
  },
  "type": [
    "Any one of:",
    "INTERNAL",
    "LDAP",
    "SSO",
    "OIDC"
  ]
}
```

Related REST command: Update the basic settings of an AuthenticationRealm.

updateAuthorizationRealm

Update the basic settings of an AuthorizationRealm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateAuthorizationRealm [parameters] [JSON file]
```

Parameters

Table 1085. Parameters

Parameter	Type	Required	Description
realm	string	true	Name or ID of the AuthorizationRealm

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "A descrption (optional)",
  "name": "AuthorizationRealm name",
  "properties": {
    "authentication-mech": "LDAP: Simple or DIGEST-MD5",
    "connection-name": "LDAP: The username to use for
  Simple binding to LDAP for searches. If not specified, an
  anonymous connection is made.",
    "connection-password": "LDAP: The password to use for
  Simple binding to LDAP for searches.",
    "group-attribute": "LDAP: The name of the attribute
  that contains group names in the user directory entry.",
    "group-base": "LDAP: The base directory to execute
  group searches in.",
    "group-name": "LDAP: The name of the entry that
  contains the user's group names in the directory entries
  returned by the group search. If this is not specified, no
  group search will take place.",
    "group-search": "LDAP: The LDAP filter expression to
  use when searching for group entries. The username will be
  put in place of {1} in the search pattern and the full
  user DN will be put in place of {0}.",
    "group-search-iterative": "LDAP: Perform additional
  lookups to find indirectly assigned groups. This option
  may significantly increase the amount of time required to
  authenticate users. Consult your LDAP administrators
  before enabling this option.",
    "group-search-subtree": "LDAP: Search the full subtree
  for the groups, as opposed to a single-level search only
  covering groups directly inside the specified search
  base.",
    "groups-delim": "SSO: A string that acts as the
  delimiter between groups that are passed in the headers.
  For example, if the user should be added to groups GROUP1
  and GROUP2, and the header value was \"GROUP=GROUP1;
  GROUP=GROUP2\" the delimiter would be ';'. Note: special
  regex characters such as '+' must be escaped with one
  backslash ''.",
    "groups-header": "SSO: The header name that denotes
  the user groups to which user will be added.",
    "groups-regex": "SSO: The regular expression to find
  the groups in the header value. If a capturing group is
  specified, only the first will be caught (per delimited
```

```
  string). If no capturing group is specified with
  parentheses, then the whole delimited string will be
  captured.",
    "sasl-realm": "LDAP: The sasl Realm for DIGEST-MD5
  authentication",
    "url": "LDAP: The URL of the LDAP server. It should
  begin with 'ldap://' or 'ldaps://'. Failover servers can
  be added by separating the URLs with a space. Example:
  ldap://ldap.mydomain.com:389 ldap://ldap.mydomain2.com",
    "user-base": "LDAP: When you search multiple
  directories, specify the starting directory that is used
  for searches, such as ou=employees,dc=mydomain,dc=com.",
    "user-pattern": "LDAP: LDAP users exist in a single
  directory; use a pattern to create the DN for users.",
    "user-search": "LDAP: The LDAP filter expression to
  use when searching for user directory entries. The
  username is put in place of {0} in the search pattern. If
  this is an attribute and not part of the user DN, wrap in
  parentheses. E.g. uid={0} or (sAMAccountName={0})",
    "user-search-subtree": "LDAP: Search the full subtree
  for the user, as opposed to a single-level search only
  covering users directly inside the specified search base.
  \"true\" or \"false\""
  }
}
```

Related REST command: Update the basic settings of an AuthorizationRealm.

updateComponent

Update the basic settings of a component

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateComponent [parameters] [JSON file]
```

Parameters

Table 1086. Parameters

Parameter	Type	Required	Description
component	string	true	The name or id of an existing component

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "cleanupCountToKeep": "Number of most recent versions to
  keep; omit this parameter to inherit the system settings
  (Optional)",
  "cleanupDaysToKeep": "Number of days to keep versions;
  omit this parameter to inherit the system settings
```

```
  (Optional)",
  "defaultVersionType": "The default type to create
  versions with; valid values are FULL and INCREMENTAL
  (Optional)",
  "description": "Description (Optional)",
  "importAutomatically": "Specify true to import new
  versions automatically (Optional)",
  "inheritSystemCleanup": "Inherit the system cleanup
  settings. If set to true, this overrides the other cleanup
  attributes. (Optional, Boolean)",
  "integrationAgent": "ID of the Agent to use for
  importing new component versions (Optional)",
  "integrationTag": "Tag name of the Agent to use for
  importing new component versions (Optional)",
  "properties": {"Source configuration plugin property
  name": "Property value"},
  "sourceConfigPlugin": "The name of the source
  configuration plug-in to use (Optional)",
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ],
  "template/nameOfTemplateProperty": "value of a template-
  inherited property with name prefixed by 'template/'
  (Optional)",
  "templateId": "ID of the template to use (Optional)",
  "templateName": "Name of a template to use; this
  parameter is ignored if  templateId is specified
  (Optional)",
  "templateVersion": "Version number of the specified
  template (Optional)",
  "useVfs": "Boolean (Optional)"
}
```

Related REST command: Update the basic settings of a component.

updateComponentResourcePropDef

Update a component resource property definition

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateComponentResourcePropDef [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "allowedValues": "Comma separated list of allowed values
  in SELECT or MULTI_SELECT properties (Optional)",
  "component": "Name or ID of the component",
  "description": "Description for the property (Optional)",
  "label": "Label of the property (Optional)",
  "name": "Name of the property to set",
  "pattern": "A regular expression that specifies valid
  values for the property. Follows java.util.regex.Pattern
  (Optional)",
  "required": "Whether the property is required (Optional,
  Boolean)",
  "type": "the type of property definition; valid values
  are TEXT (the default), , TEXTAREA, CHECKBOX, SELECT,
  MULTI_SELECT, DATETIME, and SECURE)",
  "value": "Default value of the property definition
  (Optional)"
}
```

Related REST command: Update a component resource property definition.

updateComponentTemplatePropDefs

Update a component template prop-def sheet from a JSON file. Replaces the property definitions.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateComponentTemplatePropDefs [parameters] [JSON file]
```

Parameters

Table 1087. Parameters

Parameter	Type	Required	Description
componentTemplate	string	true	Name or ID of the component template

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
```

```
    "allowedValues": "Optional, array of elements {\"
  value\":\"Required, a value\",\"label\":\"Optional,
  defaults to value\"}",
    "description": "Optional",
    "httpAuthenticationType": "Optional, BASIC or BEARER",
    "httpBasePath": "Optional",
    "httpFormat": "JSON or XML",
    "httpLabelPath": "Optional",
    "httpPassword": "Optional",
    "httpUrl": "Optional, required for Http property types",
    "httpUsername": "Optional",
    "httpValuePath": "Optional",
    "label": "Optional, label displayed to user",
    "name": "Required, the property name",
    "pattern": "Optional, regular expression for the value",
    "required": "Optional boolean, if the property is
  required",
    "resolveHttpValuesUrl": "Optional",
    "type": "Optional, Default TEXT, Type of property
  definition",
    "value": "Optional, Default value"
  }]
}
```

Related REST command: Update a component template prop-def sheet from a JSON file.

updateEnvironment

Update an environment

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateEnvironment [parameters] [JSON file]
```

Parameters

Table 1088. Parameters

Parameter	Type	Required	Description
environment	string	true	Name or ID of the environment
application	string	false	Name or ID of the application; this value is required if you specify the environment name instead of ID

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationId": "Application ID associated with the
  Environment template.",
```

```
  "cleanupCountToKeep": "Number of most recently deployed
  versions to keep (optional)",
  "cleanupDaysToKeep": "Number of days to keep versions
  deployed to this environment (optional)",
  "color": "HTML color code for the environment (optional)",
  "description": "Description (optional)",
  "exemptProcessesArray": "An array of process Ids. These
  processes will be exempt from approvals for this
  environment (optional).",
  "externalApprovalAgent": "Name or id of agent to run
  external approval processes on (optional).",
  "externalApprovalAgentPool": "Name or id of agent pool
  to run external approval processes on (optional).",
  "externalApprovalProcess": "Name or id of external
  approval process(optional).",
  "historyCleanupDaysToKeep": "Number of days to keep
  application process history for this environment
  (optional)",
  "inheritSystemCleanup": "Inherit the system cleanup
  settings. If set to true, this overrides the other cleanup
  attributes. (optional, boolean)",
  "lockSnapshots": "Whether snapshots will be locked when
  deployed to this environment (optional)",
  "name": "Environment name",
  "noSelfApprovals": "When this option is selected, users
  that submit deployment requests cannot approve their own
  requests (optional)",
  "requireApprovals": "Whether approvals will be required
  (optional)",
  "requireSnapshot": "If enabled, all deployments must use
  a snapshot instead of loose versions (optional)",
  "snapshotDaysToKeep": "Number of days to keep snapshots
  deployed to this environment (optional)",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }],
  "useSystemDefaultDays": "Whether or not to use the
  system default number of days to keep application process
  history for this environment (optional)"
}
```

Related REST command: Update an environment.

updateEnvironmentProperty

Update a component environment property

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateEnvironmentProperty [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "component": "Name or ID of the component",
  "default": "Default value (Optional)",
  "description": "Description for the property (Optional)",
  "label": "Label of the property (Optional)",
  "name": "Name of the property",
  "pattern": "A regular expression that specifies valid
  values for the property. Follows java.util.regex.Pattern
  (Optional)",
  "required": "Whether the property is required (Optional,
  Boolean)"
}
```

Related REST command: Update a component environment property.

updateEnvironmentTemplate

Update the basic settings of an environment template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateEnvironmentTemplate [parameters] [JSON file]
```

Parameters

Table 1089. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	ID of the environment template

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "applicationTemplateId": "ID of the application template
  to which the environment template belongs",
  "cleanupCountToKeep": "Number of most recently deployed
  versions to keep (Optional)",
```

```
  "cleanupDaysToKeep": "Number of days to keep versions
  deployed to this environment (Optional)",
  "color": "HTML color code for the environment template",
  "description": "description of the template (Optional)",
  "environmentType": "type of environment, leave blank for
  Standard Environment",
  "exemptProcessesArray": "An array of process Ids. These
  processes will be exempt from approvals for this
  environment (Optional).",
  "externalApprovalAgent": "Name or id of agent to run
  external approval processes on (Optional).",
  "externalApprovalAgentPool": "Name or id of agent pool
  to run external approval processes on (Optional).",
  "externalApprovalProcess": "Name or id of external
  approval process(Optional).",
  "historyCleanupDaysToKeep": "Number of days to keep
  application process history for this environment
  (Optional)",
  "lockSnapshots": "Whether snapshots will be locked when
  deployed to this environment (Optional)",
  "name": "name of the environment template",
  "noSelfApprovals": "When this option is selected, users
  that submit deployment requests cannot approve their own
  requests (Optional)",
  "requireApprovals": "Whether approvals will be required
  (Optional)",
  "requireSnapshot": "If enabled, all deployments must use
  a snapshot instead of loose versions (Optional)",
  "resourceTemplate": "ID of the resource template to use",
  "snapshotDaysToKeep": "Number of days to keep snapshots
  deployed to this environment (Optional)",
  "snapshotLockType": "type of snapshot lock: ALL,
  VERSIONS, CONFIGURATION (Required with \"lockSnapshots\"",
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (Optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (Optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ],
  "useSystemDefaultDays": "Whether or not to use the
  system default number of days to keep application process
  history for this environment (Optional)"
}
```

Related REST command: Update the basic settings of an environment template.

updateEnvironmentTemplatePropDefs

Update an environment template prop sheet def from a JSON file.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateEnvironmentTemplatePropDefs [parameters] [JSON file]
```

Parameters

Table 1090. Parameters

Parameter	Type	Required	Description
environmentTemplate	string	true	ID of the environment template

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "README": "See entity attribute for the template
  description expected by this endpoint.",
  "entity": [{
    "allowedValues": "Optional, array of elements {\"
  value\":\"Required, a value\",\"label\":\"Optional,
  defaults to value\"}",
    "description": "Optional",
    "httpAuthenticationType": "Optional, BASIC or BEARER",
    "httpBasePath": "Optional",
    "httpFormat": "JSON or XML",
    "httpLabelPath": "Optional",
    "httpPassword": "Optional",
    "httpUrl": "Optional, required for Http property types",
    "httpUsername": "Optional",
    "httpValuePath": "Optional",
    "label": "Optional, label displayed to user",
    "name": "Required, the property name",
    "pattern": "Optional, regular expression for the value",
    "required": "Optional boolean, if the property is
  required",
    "resolveHttpValuesUrl": "Optional",
    "type": "Optional, Default TEXT, Type of property
  definition",
    "value": "Optional, Default value"
  }]
}
```

Related REST command: Update an environment template prop sheet def from a JSON file..

updateExternalApprovalProcess

Update the basic settings of an external approval process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateExternalApprovalProcess [parameters] [JSON file]
```

Parameters

Table 1091. Parameters

Parameter	Type	Required	Description
externalApprovalProcess	string	true	Name or ID of the external approval process

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Description of the process",
  "name": "Name of the process",
  "propDefs": ["(Optional)"],
  "teamMappings": [
    {
      "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (Optional, can specify either this or
  resourceRoleLabel",
      "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (Optional, can specify either this or
  resourceRoleId",
      "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
      "teamLabel": "Name of the team to map the
  environment to, either this field or teamId are needed to
  specify a team"
    },
    "(Optional)"
  ],
  "workingDir": "The working directory of the process"
}
```

Related REST command: Update the basic settings of an external approval process.

updateMaintenanceBlackout

Modifies the given maintenance blackout. A date cannot be modified if it has already occurred. For recurring blackouts, only real events (synthetic=false) can be updated. For recurring blackouts, updating the date of the real event also alters the start and duration of future events within each recurrence time interval (day, week, or month).

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateMaintenanceBlackout [parameters]
```

Parameters

Table 1092. Parameters

Parameter	Type	Required	Description
id	string	true	The id of the maintenance blackout to cancel.
startDate	string	false	(Optional) The date the maintenance blackout starts. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time. (Optional)
endDate	string	false	(Optional) The date the maintenance blackout ends. In the format MM-dd-yyyy-HH:mm or in milliseconds since the UNIX epoch time. (Optional)
name	string	false	(Optional) The name or description of the blackout.

Related REST command: Modify the given maintenance blackout.

updateResource

Update a resource

To update a resource with this command, you must specify all of the existing data for the resource.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateResource [parameters] [JSON file]
```

Parameters

Table 1093. Parameters

Parameter	Type	Required	Description
resource	string	true	ID or path to the resource, such as /ResourceGroup/Agents/Agent1
template	string	false	Name or ID of the resource template (Optional)
blueprintId	string	false	Name or ID of the blueprint (Optional)
application	string	false	Name or ID of the application (Required to look up blueprint by name)

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "agent": "Name or ID of an agent to use for automation
  (Optional)",
  "agentPool": "Name or ID of an agent pool to use for
  automation (Optional)",
  "component": "Name or ID of a component to associate
  with the resource (Optional)",
  "componentTag": "Name or ID of a component tag to
  associate with the resource; any components with that tag
  can be deployed to this resource (Optional)",
  "description": "Description",
  "impersonationForce": "Specify true to prevent children
  or steps from specifying their own impersonation settings
  (Optional)",
  "impersonationGroup": "Group for default impersonation
  (Optional)",
  "impersonationPassword": "Password for default
  impersonation. Agents running on Linux and Unix machines
  will ignore this value (Optional)",
  "impersonationUseSudo": "Specify true to use the sudo
  command for impersonation. Agents running on Windows
  machines will ignore this value (Optional)",
  "impersonationUser": "User name for default
  impersonation (Optional)",
  "inheritTeam": "Whether or not to set inherit teams for
  resource. teamMappings will be ingored if true. Defaults
  to true (Optional)",
  "name": "Name for the new resource; leave blank to
  inherit the name from an agent, pool, or role",
  "parent": "ID or path to parent resource; leave blank to
  create a root resource (Optional)",
  "role": "Name or ID of a resource role to associate with
  the agent; if this resource is for a component, specify
  the component name instead (Optional)",
  "roleProperties": {"Property name": "Property value
  (Optional)"},
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  }]
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  updateResource
  -resource "/My resource"
  updateResource.json
```

Example

Example JSON request

```
{
  "name": "New resource name",
  "description": "An updated resource",
}
```

Related REST command: Update a resource.

updateResourceTemplate

Update a resource template from a JSON file

To update a resource template with this command, you must specify all of the existing data for the resource template as defined in the JSON data template, except 'teamMappings' which is optional.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateResourceTemplate [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Description",
  "existingId": "ID of resource template; leave blank when
  creating new template",
  "name": "Name for the resource template",
  "rootTeamMappings": "Array of team ids; if not
  specified, no teams will be mapped to Root Resource;
  format: [ {teamId: '<ID-of-team-1>'}, {teamId: '<ID-of-
  team-2>'}, ... ]; (Optional)",
  "teamMappings": "Array of team ids; if not specified, no
  teams will be mapped to Template; format: [ {teamId: '<ID-
  of-team-1>'}, {teamId: '<ID-of-team-2>'}, ... ];
  (Optional)"
}
```

Related REST command: Update a resource template from a JSON file.

updateRoleActions

Update any number of Actions for one or more Roles + Types

Update the Actions for one or more Roles. Only explicitly specified mappings will be added/removed.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateRoleActions [JSON file]
```

Parameters

None.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{"[Role Name]": {"[Object Type]": {"[Type Name]":
  {"[Action]": "true|false"}}}}
```

Related REST command: Update any number of Actions for one or more Roles + Types.

updateTeam

Update a team

Specify the team name in the 'team' parameter and use the 'name' and 'description' parameters to specify the desired changes.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateTeam [parameters]
```

Parameters

Table 1094. Parameters

Parameter	Type	Required	Description
team	string	true	Name of the team to be changed
name	string	false	New name of the team
description	string	false	New description of the team

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
```

```
  updateTeam
  -team MyNewTeam
  -name ADifferentTeamName
  -description "ADifferentTeamDescription"
```

Related REST command: Update a team.

updateUser

Update a user

Specify the user name in the 'user' parameter and use the JSON string to specify new information about the user. This command can update the name and email address of the user, but not the password.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateUser [parameters] [JSON file]
```

Parameters

Table 1095. Parameters

Parameter	Type	Required	Description
user	string	true	Name/ID of the user. The name will be case-sensitive if a case-sensitive realm is provided.
realm	string	false	Limit any user name search to given AuthenticationRealm.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "actualName": "User real name (optional)",
  "email": "Email address (optional)"
}
```

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  updateUser
  -user jsmith
  updateUser.json
```

Example

Example JSON request

```
{
  "password": "password",
  "actualName": "Jane Smith",
  "email": "jsmith@example.org"
}
```

Related REST command: Update a user.

updateUsersInAuthRealm

Update all users in an authentication realm

This command updates all users in an authentication realm

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateUsersInAuthRealm
```

Parameters

Table 1096. Parameters

Parameter	Type	Required	Description
authenticationRealm	string	true	Name of the authentication realm

Related REST command: Update all users in an authentication realm.

updateVaultAppRole

Update a Vault AppRole

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateVaultAppRole [parameters] [JSON file]
```

Parameters

Table 1097. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultAppRole	string	true	Name or ID of the Vault App Role

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "The Vault App Role Name (Optional, Unique per
  Vault store)",
  "roleId": "The Vault Role ID (Optional)",
  "secretId": "The Vault Secret ID (Optional)"
}
```

Related REST command: Update a Vault AppRole.

updateVaultCertificate

Update a Vault Certificate

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateVaultCertificate [parameters] [JSON file]
```

Parameters

Table 1098. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Secret Store ID/Name
vaultCertificate	string	true	Name or ID of the Vault Certificate

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "certificatePEM": "The certificate to be stored in PEM
  format(Optional - required if privateKeyPEM set)",
  "name": "The Vault Certificate Name (Optional, Unique
  per Vault store)",
  "privateKeyPEM": "The private key to be stored in PEM
  format(Optional - required if certificatePEM set)"
}
```

Related REST command: Update a Vault Certificate.

updateVaultLdapUser

Update a new Vault Ldap User

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateVaultLdapUser [parameters] [JSON file]
```

Parameters

Table 1099. Parameters

Parameter	Type	Required	Description
store	string	false	Vault Store ID/Name
vaultLdapUser	string	true	Name or ID of the Vault Ldap User

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "name": "The Vault Ldap User Name (Required, Unique per
  Vault store)",
  "password": "The Vault Ldap Users password (Optional)"
}
```

Related REST command: Update a new Vault Ldap User.

updateVaultSecretStore

Update a secret store.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateVaultSecretStore [parameters] [JSON file]
```

Parameters

Table 1100. Parameters

Parameter	Type	Required	Description
store	string	true	Name or id of the secret store.

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "description": "Secret Store description",
```

```
  "name": "Secret Store name",
  "namespace": "Vault Server Namespace (optional)",
  "teamMappings": {
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the environment
  to, either this field or teamId are needed to specify a
  team"
  },
  "type": "vault",
  "url": "Vault Server URL"
}
```

Related REST command: Update a secret store..

updateWebhookForApplication

Update an application webhook

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateWebhookForApplication [parameters]
```

Parameters

Table 1101. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	false	ID or name of webhook template
url	string	false	Url for the webhook
environment	string	false	ID of environment to add the webhook to. This can be repeated for multiple environments.
webhookProxyHost	string	false	Host name of http proxy.
webhookProxyPort	string	false	Port of http proxy.
webhookProxyUsername	string	false	Username for authentication with http proxy.

Table 1101. Parameters (continued)

Parameter	Type	Required	Description
webhookProxyPassword	string	false	Password for authentication with http proxy.
webhook	string	true	ID of webhook
application	string	true	ID or name of application

Related REST command: Update an application webhook.

updateWebhookForApplicationWithCredentials

Update an application webhook

Update the settings for an application webhook.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateWebhookForApplicationWithCredentials [JSON file]
```

Parameters

Table 1102. Parameters

Parameter	Type	Required	Description
webhook	string	true	ID of webhook
application	string	true	ID or name of application

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "environments": ["Name or ID of environments"],
  "proxyHost": "Proxy Host. Updating proxy host will unset
  proxy username and password if those are not provided",
  "proxyPassword": "Password to use for basic
  authentication with the proxy",
  "proxyPort": "Proxy Port. Updating proxy port will unset
  proxy username and password if those are not provided",
  "proxyUsername": "Username to use for basic
  authentication with the proxy",
  "url": "Url for the webhook",
  "webhookTemplate": "Name or ID of the webhook template"
```

```
}
```

Related REST command: Update an application webhook.

updateWebhookForProcess

Updates a webhook for a process

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateWebhookForProcess [parameters]
```

Parameters

Table 1103. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	Id or name of webhook template.
url	string	true	Url for the webhook
webhookProxyHost	string	false	Host name of http proxy.
webhookProxyPort	string	false	Port of http proxy.
webhookProxyUsername	string	false	Username for authentication with http proxy..
webhookProxyPassword	string	false	Password for authentication with http proxy.
process	string	true	Id or name of process
webhook	string	true	Id of webhook

Related REST command: Updates a webhook for a process.

updateWebhookForProcessWithCredentials

Updates a webhook for a process.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateWebhookForProcessWithCredentials [JSON file]
```

Parameters

Table 1104. Parameters

Parameter	Type	Required	Description
process	string	true	Id or name of process
webhook	string	true	Id of webhook

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "proxyHost": "Proxy Host. Updating proxy host will unset
  proxy username and password if those are not provided",
  "proxyPassword": "Password to use for basic
  authentication with the proxy",
  "proxyPort": "Proxy Port. Updating proxy port will unset
  proxy username and password if those are not provided",
  "proxyUsername": "Username to use for basic
  authentication with the proxy",
  "url": "Url for the webhook",
  "webhookTemplate": "Name or ID of the webhook template"
}
```

Related REST command: Updates a webhook for a process..

updateWebhookTemplate

Update a webhook template

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  updateWebhookTemplate [parameters] [JSON file]
```

Parameters

Table 1105. Parameters

Parameter	Type	Required	Description
webhookTemplate	string	true	Name or ID of the webhook template to update

Template

This command takes a JSON request string or file. Use the following template for the request:

```
{
  "body": "The webhook template body specified with Apache
  Velocity template syntax",
  "description": "Description (Optional)",
  "id": "ID of the webhook template to update",
  "name": "Name of the new webhook template",
  "teamMappings": [{
    "resourceRoleId": "Id of Resource Type to apply to
  this team mapping (optional, can specify either this or
  resourceRoleLabel",
    "resourceRoleLabel": "Name of Resource Type to apply
  to this team mapping (optional, can specify either this or
  resourceRoleId",
    "teamId": "ID of a team, either this field or
  teamLabel are needed to specify a team",
    "teamLabel": "Name of the team to map the webhook
  template to, either this field or teamId are needed to
  specify a team"
  }],
  "type": "Webhook template type [PROCESS_SUCCESS,
  PROCESS_FAILURE, PROCESS_STARTED, APPROVAL_COMPLETED,
  APPROVAL_FAILED, EXTERNAL_APPROVAL_COMPLETED,
  EXTERNAL_APPROVAL_FAILED, SCHEDULED_PROCESS_NOT_STARTED,
  TASK_CREATED]"
}
```

Related REST command: Update a webhook template.

upgradeAgent

Upgrade an agent

The agent will be upgraded to the agent version corresponding with the server version.

Format

```
udclient [global-args...] on page 2592 [global-flags...] on page 2592
  upgradeAgent [parameters]
```

Parameters

Table 1106. Parameters

Parameter	Type	Required	Description
agent	string	true	Name or ID of the agent

Example

Example

```
udclient -username jsmith -password passwd
  -weburl https://deployserver.example.com:8443
  upgradeAgent
  -agent Agent1
```

Related REST command: Upgrade an agent.

Process-as-code feature

Programmatically manage processes with PAC language.

The process as code feature (PAC) lets you use the command line to create HCL DevOps Deploy (Deploy) processes. You can create scripts that define processes and then upload them to the server. You can download process designs and modify them in your text editor.

The PAC language supports Deploy processes. You write PAC scripts in the PAC language. The PAC language provides elements that describe process step commands, step properties, and connecting steps, such as Switch. PAC files are identified by the `pac` file extension.

Using the PAC commands

Use the PAC language to create HCL DevOps Deploy (Deploy) processes.

Before you begin

To get started, install the PAC tools.

1. On the dashboard, click **Help > Tools > Process-as-code Compiler**.
2. Extract the downloaded `pacc-version_number` file.
3. Optionally, if you want to reference PAC commands from anywhere on your computer, add the folder to your `path` system variable.

About this task

To use PAC scripts in your applications, upload valid PAC files to the Deploy server. You initially create processes in the process designer and then download them as PAC files. When you finish editing the PAC script, you upload them to the server. The processes that you define in PAC files are identical to the processes that you define in the process designer.

The PAC tools consist of a number of command files. The primary commands are upload and download. The primary commands are further organized by type: application, component, and generic. For example, the PAC command to upload an application process is: `upload-application-process`. The PAC command to download a generic process is: `download-generic-process`. Similarly, you can upload and download a component process using the PAC commands.

Upload examples

The command format for uploading PAC files is provided in the following code example:

```
upload-<process-type>-process <user-name> <server-url> <process-id> <next-version> input.pac
```

All PAC scripts have the same arguments and differ only by the type of process referenced. For example, to upload an application-type file named `my_command.pac`, the command might look like the following code example:

```
upload-application-process my_user_name https://my_server:4540 17542df8-a0a3-7b7d-ffa9-88ebe19324a9 2
 my_command.pac
```

This command example uploads a new version of the application process defined in the `my_command.pac` file.

Download examples

The command format for downloading PAC files is provided in the following code example:

```
download-<process-type>-process <user-name> <server-url> <process-id> <next-version> input.pac
```

For example, to download an application-type file named `my_command.pac`, the command might look like the following code example:

```
download-application-process my_user_name https://my_server:4540 17542df8-a0a3-7b7d-ffa9-88ebe19324a9 3
 my_command.pac
```

This command example downloads the process identified by the version ID to a PAC file named `my_command.pac`. If the version is not specified, the latest version is downloaded.

Silent upload and download of PAC files

The upload and download commands prompt you to enter your password manually. However, you can upload and download the PAC files silently by adding the following parameters to the commands:

- `--password <your_PAC_password>`

  For example, the following command uploads the PAC file silently:

  ```
  upload-<process-type>-process <user-name> <server-url> <process-id> <next-version> input.pac --password
   <your_PAC_password>
  ```

- `--password-variable MY_PAC_PASSWORD`, where MY_PAC_PASSWORD is an environment variable that defines your PAC password. You can use this option in automation and scripts to supply the password without exposing it directly in the commands.

  For example, the following command uploads the PAC file without exposing your password in the command:

  ```
  upload-<process-type>-process <user-name> <server-url> <process-id> <next-version> input.pac
   --password-variable MY_PAC_PASSWORD
  ```

Notes:

- If you use both parameters in a command, the `--password` parameter is used in preference to the `--password-variable` parameter.
- If you use `--password-variable` but the named environment variable is not set, then the parameter does not work. In this case, the console prompts you to provide the password manually.

To implement PAC processes, complete the following steps:

1. From the dashboard, create a process.
   The process can be of any type, application, component, or generic. You use the ID assigned by the server to the process in your PAC files to identify the process. You do not need to define steps with the process designer.
2. From the command line, download the PAC file using the `download-<process-type>-process` PAC command.
3. In any text editor, edit the PAC file.
4. Upload the PAC file using the process ID assigned to it by the server.
   Along with the process ID you obtained from the server, you must assign a unique process version number to the uploaded process.

Results

The next time that you open the process designer, the designer displays the new process defined in the PAC file.

Understanding the PAC file format

Overall Syntax

```
-- Comments are indicated by a lead pair of dashes
```

The `start` block defines which steps will begin the process. Multiple steps can be started concurrently by listing multiple starts

```
start is
    start "step-1"
    start "step-2"
end
```

Step blocks define steps and begin with a keyword for the kind of step and declare the name of the step. Steps are linked together by name, so every step must have a unique name.

```
plugin step "step-1" is
    ...   -- explained later
end
```

Steps define what action to take after the step ends depending on the outcome.

```
plugin step "step-1" is
    ...
on success
    -- actions when the step succeeds
    start "step-2"
on failure
    -- actions when the step fails
    start "step-3"
on complete
    -- actions when the step completes regardless of success or failure
    start "step-4"
end
```

Each outcome can specify multiple actions. The action 'finish' means to start the implicit finish step.

```
plugin step "step-1" is
    ...
on success
```

```
    start "step-2"
    start "step-3"
    start "step-4"
on failure
    finish
```

Certain steps allow nesting processes.

```
for-each-resource-tag step "frt1" is
    ... -- explained later

    start is
        start "step-1"
    end

    plugin step "step-1" is
        ...
    on success
        finish
    end
on success
    start "step-2"
end
```

Step names must unique across top-level and nested processes. Nested process steps may only reference their nest mates.

String Syntax

Strings are used to specify step names and other custom values. Strings are specified by text surrounded by double quote characters.

```
"single quote"
```

Special characters can be embedded by escaping with a backslash. Double quote, carriage return, new line, horizontal tab, and backslash itself may be escaped this way.

```
" \" \r \n \t \\ "
```

Strings can also be specified by triple quotes. These strings do not allow escape characters, but they can extend over multiple lines or embed other special characters without quoting. When using triple quotes, triple quotes must be placed on newline. This restriction may be lifted later.

```
"""
cd C:\workdir
copy *.* "C:\Program Files\My Program"
"""
```

Triple quotes can be embedded in a triple quote string by doubling.

```
""" """""" <- becomes " repeated 3 times """
```

Multi-line triple quoted strings have the leading indent removed, where the indent is determined by the identation of the opening triple quote.

For example,

```
    """
    for x in *.txt
    do
        echo $x
        cat $x
    done
    """
```

becomes

```
for x in *.txt
do
    echo $x
    cat $x
done
```

Implemented Step Types

These are the currently implemented step types. Refer to the main product documentation for more information about the behavior of each step in a process.

Any type of process can be defined with this language, but not all steps are valid in every type of process.

`start`

The `start` block is a pseudo-step that defines the beginning of the process. As it is not a true step, it does not use the `step` keyword. The only completion condition is success, so `on success` is not required or permitted.

```
start is
    -- list one or more actions
    start "foo"
    start "bar"
end
```

Application process steps

Lists the various application steps that can be used in a process.

application-run-component-process

The `application-run-component-process` step executes a component process and waits for it to complete. This step is only valid in an application process.

Format

```
application-run-component-process step "foo" is
    -- name of the component to deploy; required
    component "my-component"

    -- name of the component process to run for each component; required
    process "deploy-my-component"

    -- select resources to run on by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "res-tag-1"
```

```
    -- select resources to run on based on inventory changes in other components; optional
    select-resources-by-changed-component-list
        -- select components to monitor for inventory changes; optional, repeatable
        -- before 7.1.1.0, the component must be specified by internal database ID
        component "comp-1"
        component "comp-2"

        -- specify which resources of this process's component are selected; required
        --   all - select all resources where this process's component is mapped
        --   with-changed-component - select only resources where both this process's component
        --                            and a monitored component are mapped and the monitored
        --                            component has changed
        select-resources all
    end

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"
on success
    finish
end
```

Table 1107.

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name
Component Process	The component process to run.

Example

Example

```
start is

    start A
```

```
    start B
    start C
    start D
end


application-run-component-process step A is
    component "comp-1"
    process "proc-1"
    select-resources-by-tag "resource-tag-1"
    select-resources-by-changed-component-list
        component "component-1"
        component "component-2"
        component "component-3"
        select-resources all
    end
    fail-fast true
    ignore-child-warnings true
    run-on-first-online-resource-only true
    max-concurrent-processes 101
    precondition-script "script-1"
    property "comp-1-prop-1" = "comp-1-prop-value-1"
    property "comp-1-prop-2" = "comp-1-prop-value-2"
    property "comp-1-prop-3" = "comp-1-prop-value-3"
on success
    finish
end
```

```
application-run-component-process step B is

    component "comp-2"

    process "proc-2"

    select-resources-by-changed-component-list

        select-resources with-changed-component

    end

on success

    finish

end



application-run-component-process step C is

    component "comp-3"

    process "proc-3"

    select-resources-by-tag "resource-tag-2"

    fail-fast true

    ignore-child-warnings true

    run-on-first-online-resource-only true

    max-concurrent-processes 102

    precondition-script "script-2"

    property "comp-3-prop-1" = "comp-3-prop-value-1"

    property "comp-3-prop-2" = "comp-3-prop-value-2"

    property "comp-3-prop-3" = "comp-3-prop-value-3"

on success

    finish

end



application-run-component-process step D is

    component "comp-4"
```

```
    process "proc-4"

on success

    finish

end
```

apply-configuration

The `apply-configuration` step starts child component processes for applying configuration changes. This step is only valid in an application process.

Format

```
apply-configuration step "foo" is
    -- name of the component to configure; required
    component "my-component"

    -- name of the component process to run; required
    -- must be a configuration deployment or a no-version-needed operational process
    process "deploy-component-process-property"

    -- select resources to apply to by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "my-tag"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"

on success
    start "bar"
end
```

Properties

Table 1108. Apply Configuration properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Maximum number of concurrent jobs	The maximum number component instances to apply the new configuration to simultaneously. For example, if you set the maximum number of concurrent processes to 2, the server applies the configuration to only two instances of a component at a time, even if many instances of the component are installed. To apply the configuration to every component instance simultaneously, specify `-1`. To limit the number of components to change at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is
    start A
    start B
end

apply-configuration step A is
    component "component-a"
    process "process-a"
    select-resources-by-tag "res-tag-a"
    fail-fast true
    ignore-child-warnings true
    max-concurrent-processes 42
    precondition-script "script-a"
    property "prop-a1" = "prop-a1-value"
    property "prop-a2" = "prop-a2-value"
    property "prop-a3" = "prop-a3-value"
on success
```

```
    finish
end

apply-configuration step B is
    component "component-b"
    process "process-b"
on success
    finish
end
```

for-each-agent

The for-each-agent application process step lets you run a process for each agent in an environment.

Format

```
for-each-agent step "loop1" is
    -- define target resource tags; optional, repeatable
    -- before 7.1.1.0, the tag must be specified by internal database ID
    tag "res-tag-1"
    tag "res-tag-2"

    -- maximum number of agents to target concurrently; optional
    -- default is unlimited
    max-concurrent-agents unlimited

    -- must contain an embedded 'start' block that works like the top-level version
    start is
        start "install-c1"
    end

    -- contains one or more embedded steps of any type except for other for-each steps
    install-component step "install-c1" is
        component "c1"
        process "deploy"
        without-status "Active"
        fail-fast true
        ignore-child-warnings true
        max-concurent-processes 43

    -- on success/failure/complete reference steps inside the for-each
    -- 'finish' refers to the implied finish step inside the for-each
    on success
        finish
    end

-- on success/failure/complete outside an embedded step reference steps in the top-level process
on success
    start "next-step"
end
```

Properties

Table 1109. For Each Agent properties

Field	Description
Name	A name for the iteration container.
Limit to Resource Tags	A list of resource tags that limit the container. The subprocess runs only on the agent resources that have one or more of the specified resource tags.
Maximum number of concurrent processes	The maximum number of agents that run the subprocess at a time. To run the subprocess on every agent at once, specify `-1`. To limit the number of agents to deploy to at once, as in blue/green deployments or rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Example

Example

```
start is
    start A
    start B
end

for-each-agent step A is
    tag "res-tag-a1"
    tag "res-tag-a2"
    tag "res-tag-a3"

    max-concurrent-agents 42

    start is
        start A1
    end

    set-final-process-status step A1 is
        status success
    on success
        finish
    end
on complete
    finish
end

for-each-agent step B is
    start is
        start B1
    end

    set-final-process-status step B1 is
```

```
        status success
    on success
        finish
    end
on complete
    finish
end
```

for-each-resource-tag

The for-each-resource-tag application process step lets you run processes for resources selected by tag. This step is only valid in an application process.

Format

The for-each-resource-tag step is called For Each Tag in the HCL DevOps Deploy (Deploy) web application.

```
for-each-resource-tag step "loop1" is
    -- define target resource tags; required, repeatable
    -- before 7.1.1.0, the tag must be specified by internal database ID
    tag "res-tag-a1"
    tag "res-tag-a2"

    -- maximum number of tags to target concurrently; optional
    -- default is 1
    max-concurrent-tags 1

    -- must contain an embedded 'start' block that works like the top-level version
    start is
        start "install-c1"
    end

    -- contains one or more embedded steps of any type except for other for-each steps
    install-component step "install-c1" is
        component "c1"
        process "deploy"
        without-status "Active"
        fail-fast true
        ignore-child-warnings true
        max-concurent-processes 43

    -- on success/failure/complete reference steps inside the for-each
    -- 'finish' refers to the implied finish step inside the for-each
    on success
        finish
    end

-- on success/failure/complete outside an embedded step reference steps in the top-level process
on success
    start "next-step"
end
```

Properties

Table 1110. For Each Tag properties

Field	Description
Name	A name for the iteration container.
Tags	A list of resource tags to use in the container. The subprocess runs once for each resource tag. You can order the resource tags by dragging and dropping them.
Max Concurrent Tags	The maximum number of tags for which to run the subprocess at a time. To run the subprocess on all of the tags at the same time, specify `-1`. To run the subprocess on only a few of the tags at a time, as in blue/green deployments or rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Example

Example

```
start is
    start A
    start B
end

for-each-resource-tag step A is
    tag "res-tag-a1"
    tag "res-tag-a2"
    tag "res-tag-a3"

    max-concurrent-tags 10

    start is
        start A1
    end

    shell step A1 is
        "echo A1"
    on success
        finish
    end
on complete
    finish
end

for-each-resource-tag step B is
    tag "res-tag-b1"

    max-concurrent-tags unlimited

    start is
        start B1
    end
```

```
    shell step B1 is
        "echo B1"
    on success
        finish
    end
on complete
    finish
end
```

install-component

The install-component application process step starts child component processes that install a single component. This step is only valid in an application process.

Format

```
install-component step "foo" is
    -- name of the component to deploy; required
    component "my-component"

    -- name of the component process to run for each component; required
    process "deploy-component-process-property"

    -- select versions for deployment without this inventory status; required
    select-versions-without-inventory-status "Active"

    -- select resources to deploy to by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "my-tag"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"

on success
```

```
    start "bar"
end
```

Properties

Table 1111. Install Component properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. To install another component, add another install step to the process.
Use Versions Without Status	Restricts the components that can be used by the step. Components with the selected status are ignored. Available statuses: `Active` means ignore components that are currently deployed.
Component Process	The component process to run.
Limit to Tag	User-defined roles can be used in processes; see Adding tags to objects on page 2521.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously to deploy the component. For example, if you set the maximum number of concurrent processes to 2, only two instances of the component are installed at a time, even if many instances of the component are mapped to agents. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is
    start A
    start B
end

install-component step A is
    component "component-a"
    process "process-a"
    select-versions-without-inventory-status "status-a"
    select-resources-by-tag "tag-a"
    fail-fast true
    ignore-child-warnings true
    max-concurrent-processes unlimited
    precondition-script "script-a"
    property "prop-a1" = "prop-a1-value"
    property "prop-a2" = "prop-a2-value"
    property "prop-a3" = "prop-a3-value"
on success
    finish
end

install-component step B is
    component "component-b"
    process "process-b"
    select-versions-without-inventory-status "status-b"
    fail-fast false
    ignore-child-warnings false
    max-concurrent-processes 10
on success
    finish
end
```

install-multiple-components

The `install-multiple-components` step runs child component processes that install multiple components. This step is only valid in an application process.

Format

```
install-multiple-components step "foo" is
    -- name of the component process to run for each component; required
    process "deploy"

    -- select versions for deployment without this inventory status; required
    select-versions-without-inventory-status "Active"

    -- select components to deploy by component tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-components-by-tag "comp-tag-a"

    -- select resources to deploy to by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "res-tag-a"
```

```
    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes per component; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- maximum number of components to deploy concurrently; optional
    -- default is 100
    max-concurrent-components 100

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

on success
    start "bar"
end
```

Properties

Table 1112. Install Multiple Components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	A process for the components that contain the selected tag. The process name must be the same for each tagged component. Only components that contain the specified process run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Use Versions Without Status	Restricts the components that can be used by the step. Components with the selected status are ignored. Available statuses: `Active` means ignore components that are currently deployed.
Maximum number of	The maximum number of components to deploy at one time. For example, if you specify 5, this step deploys five components at a time to all agents to which the components are mapped. To run this step on all

Table 1112. Install Multiple Components properties (continued)

Field	Description
concurrent components	components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to deploy each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are installed at a time, even if many instances of the component are mapped to agents. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is
    start A
    start B
end

install-multiple-components step A is
    process "process-a"
    select-versions-without-inventory-status "status-a"
    select-components-by-tag "comp-tag-a"
    select-resources-by-tag "res-tag-a"
    fail-fast true
    ignore-child-warnings true
```

```
    max-concurrent-processes unlimited
    max-concurrent-components unlimited
    precondition-script "script-a"
on success
    finish
end

install-multiple-components step B is
    process "process-b"
    select-versions-without-inventory-status "status-b"
    max-concurrent-processes 1
    max-concurrent-components 2
on success
    finish
end
```

join

The join application process step.

Format

The `join` step halts processing until all connected steps are completed. The only valid completion condition is success, so `on success` is not required or permitted.

This step does not have properties.

Example

Example

```
join step "foo" is
    finish
end
```

release-lock step

The release-lock step releases a process lock.

Format

```
join step "foo" is
    finish
end
```

Properties

Table 1113. Release lock properties

Field	Description
Lock Name	A string value that identifies the lock.

Example

Example

```
start is
    start "A"
end

release-lock step "A" is
    lock "lock-name"
on success
    finish
end
```

rollback-component step

The rollback-component step starts child component processes for rolling back a single component. This step is only valid in an application process.

Format

```
rollback-component step "foo" is
    -- name of the component to deploy; required
    component "my-component"

    -- name of the component process to run for each component; required
    process "deploy-component-process-property"

    -- mode for selecting versions to rollback; required
    --   remove-undesired-incremental-versions - rollback incremental versions
    --   replace-with-last-deployed - rollback to previous version
    rollback-type replace-with-last-deployed

    -- select versions for rollback with this inventory status; required
    select-versions-with-inventory-status "Active"

    -- select resources to rollback on by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "my-tag"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
```

```
        true
        """

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"

on success
    start "bar"
end
```

Properties

Table 1114. Rollback Component properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. If you want to roll back another component, add another rollback step to the process or use the Rollback Multiple Components on page 2486 step.
Remove Versions with Status	Restricts the component versions that are affected by the step. Only versions with the selected status are affected. By default, the status `Active` refers to versions that are currently deployed.
Component Process	The component process to run. In most cases, select a component process that uninstalls the component.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Rollback type	Determines the type of rollback. If you specify `Remove Undesired Incremental Versions`, the server runs the uninstall process for each version that is not in the specified snapshot. See Rolling back to a snapshot on page 2437. If you specify `Replace with Last Deployed`, the server runs the specified process on the version that was installed when the process began. See Uninstalling components with an application process on page 2436.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are rolled back at a time, even if there are many instances of the component. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Table 1114. Rollback Component properties (continued)

Field	Description
	Note: The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is
    start A
    start B
end

rollback-component step A is
    component "component-a"
    process "process-a"
    rollback-type replace-with-last-deployed
    select-versions-with-inventory-status "status-a"
    select-resources-by-tag "tag-a"
    fail-fast true
    ignore-child-warnings true
    max-concurrent-processes 100
    precondition-script "script-a"
    property "prop-a1" = "prop-a1-value"
    property "prop-a2" = "prop-a2-value"
    property "prop-a3" = "prop-a3-value"
on success
    finish
end

rollback-component step B is
    component "component-b"
    process "process-b"
    rollback-type remove-undesired-incremental-versions
    select-versions-with-inventory-status "status-b"
on success
    finish
end
```

rollback-multiple-components step

The rollback-multiple-components step starts child component processes for rolling back multiple components. This step is only valid in an application process.

Format

```
rollback-multiple-components step "foo" is
    -- name of the component process to run for each component; required
    process "deploy"

    -- mode for selecting versions to rollback; required
    --   remove-undesired-incremental-versions - rollback incremental versions
    --   replace-with-last-deployed - rollback to previous version
    rollback-type replace-with-last-deployed

    -- select versions for rollback with this inventory status; required
    select-versions-with-inventory-status "Active"

    -- select components to rollback by component tag; optional
    -- currently, the tag must be specified by internal database ID
    select-components-by-tag "1714f9ea-1aa0-405d-f8df-ae131fcee7be"

    -- select resources to rollback by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "my-tag"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes per component; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- maximum number of components to uninstall concurrently; optional
    -- default is 100
    max-concurrent-components 100

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

on success
    start "bar"
end
```

Properties

Table 1115. rollback-multiple-components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.

Table 1115. rollback-multiple-components properties (continued)

Field	Description
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	The component process to run. In most cases, select a component process that uninstalls the component.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Remove Versions With Status	Restricts the components that are affected by the step. Only components with the selected status are affected. Available statuses: `Active` refers to components that are currently deployed.
Rollback Type	Determines the type of rollback. If you specify `Remove Undesired Incremental Versions`, the server runs the uninstall process for each version that is not in the specified snapshot. See Rolling back to a snapshot on page 2437. If you specify `Replace with Last Deployed`, the server runs the specified process on the version that was installed when the process began. See Uninstalling components with an application process on page 2436.
Maximum number of concurrent components	The maximum number of components to roll back at one time. For example, if you specify 5, this step rolls back five components at a time on all agents to which the components are installed. To run this step on all components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to roll back each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are rolled back at a time, even if there are many instances of the component. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer.

Table 1115. rollback-multiple-components properties (continued)

Field	Description
	The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is
    start A
    start B
end

uninstall-multiple-components step A is
    process "process-a"
    select-versions all
    select-versions-with-inventory-status "status-a"
    select-components-by-tag "comp-tag-a"
    select-resources-by-tag "res-tag-a"
    fail-fast true
    ignore-child-warnings true
    max-concurrent-processes 42
    max-concurrent-components 24
    precondition-script "script-a"
on success
    finish
end

uninstall-multiple-components step B is
    process "process-b"
    select-versions selected-with-process
    select-versions-with-inventory-status "status-b"
on success
    finish
end
```

run-operational-process-for-multiple-components

The `run-operational-process-for-multiple-components` step executes a component operational process for multiple components and waits for them to complete. This step is only valid in an application process.

Format

```
run-operational-process-for-multiple-components "foo" is
    -- name of the component process to run for each component; required
    process "my-component-operation"

    -- select components to target by component tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-components-by-tag "my-tag"

    -- select resources to run on by resource tag; optional
    -- currently, the tag must be specified by internal database ID
    select-resources-by-tag "171603dd-389d-3183-c496-5ca60f4378f2"

    -- select resources to run on based on inventory changes in other components; optional
    select-resources-by-changed-component-tag
        -- select components to monitor for inventory changes by tag; optional
        -- currently, the tag must be specified by internal database ID
        select-components-by-tag "1714f9ea-1aa0-405d-f8df-ae131fcee7be"

        -- specify which resources of this process's component are selected; required
        --   all - select all resources where this process's component is mapped
        --   with-changed-component - select only resources where both this process's component
        --                            and a monitored component are mapped and the monitored
        --                            component has changed
        select-resources all
    end

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- only run on one online resource; optional
    -- default is false
    run-on-first-online-resource-only true

    -- maximum number of concurrent processes per component; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- maximum number of components to deploy concurrently; optional
    -- default is 100
    max-concurrent-components 100

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
```

```
        """

on success
    finish
end
```

Table 1116. Operational component process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	A process for the components that contain the selected tag. Only components that contain the process will run. Only one process can be selected per step.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Maximum number of concurrent co	The maximum number of components on which to run processes at one time. For example, if you specify 5, this step runs processes on five components at a time. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Table 1116. Operational component process properties (continued)

Field	Description
mponents	
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously. For example, if you set the maximum number of concurrent processes to 2, the process runs on only two instances of a component at a time, even if many instances of the component are installed. To run an unlimited number of concurrent processes, specify `-1` . To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Run on First Online Resource Only	Instead of being run by all agents that are mapped to the environment, the step runs only on the first online agent that the server identifies. The mechanism that is used to identify the "first" agent is database-dependent, and thus indeterminate. If no agents are online, the server skips the step and marks its status as "Not Mapped." This setting may cause the environment to become noncompliant or to provide unexpected results, because only one agent runs the process, even if many agents are mapped to the environment.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Run If Co	To run this step only if another step in the application process changes the inventory of a component, select this option. If you select a component tag, for this step to run, a component that contains that tag must have an inventory change. If you do not select a component tag, any inventory change that is caused by the application process causes

Table 1116. Operational component process properties (continued)

Field	Description
mponentwithTagChanges	this step to run. **Note:** Do not select the same component tag that you selected in the **Component Tag** list. Select an option in the **Resource Selection Mode** list to control the agent resources that this step runs on. To run the step on only resources where the inventory of other components changes, select **Only resources where component with selected tag is changed**. For example, you might have a component that contains some operational processes, such as "Restart HTTP Server." If you add the operational process to the application process, you can select this option to run only on machines where you are deploying changes to websites that are hosted. To run the operational process on all possible resources, select **All resources mapped to this process's component**. For example, if you deploy changes to a website, you might want to restart a load balancer that is on a different machine than the website. You can select this option to run the operational process wherever the "Load Balancer" component that contains the "Restart Load Balancer" operational process is mapped.

Example

Example

```
start is

    start A

    start B

    start C

    start D

    start E

end



run-operational-process-for-multiple-components step A is

    process "proc-1"

    select-components-by-tag "comp-tag-1"

    select-resources-by-tag "resource-tag-1"

    select-resources-by-changed-component-tag

        select-components-by-tag "comp-tag-2"

        select-resources all

    end
```

```
    fail-fast true

    ignore-child-warnings true

    run-on-first-online-resource-only true

    max-concurrent-processes 1

    max-concurrent-components 2

    precondition-script "script-1"

on success

    finish

end



run-operational-process-for-multiple-components step B is

    process "proc-2"

    select-resources-by-changed-component-tag

        select-resources with-changed-component

    end

on success

    finish

end



run-operational-process-for-multiple-components step C is

    process "proc-3"

    select-resources-by-changed-component-tag

        select-components-by-tag "comp-tag-3"

        select-resources with-changed-component

    end

on success

    finish

end
```

```
run-operational-process-for-multiple-components step D is
    process "proc-4"
    select-components-by-tag "comp-tag-4"
    select-resources-by-tag "resource-tag-2"
    fail-fast true
    ignore-child-warnings true
    run-on-first-online-resource-only true
    max-concurrent-processes 3
    max-concurrent-components 4
    precondition-script "script-2"
on success
    finish
end

run-operational-process-for-multiple-components step E is
    process "proc-5"
on success
    finish
end
```

uninstall-component

The `uninstall-component` step starts child component processes for uninstalling a single component. This step is only valid in an application process.

Format

```
uninstall-component step "foo" is
    -- name of the component to deploy; required
    component "my-component"

    -- name of the component process to run for each component; required
    process "deploy-component-process-property"
```

```
    -- mode for selecting versions to uninstall; required
    --   all - uninstall all versions installed
    --   selected-with-process - uninstall versions selected with the process
    select-versions all

    -- select versions for deployment with this inventory status; required
    select-versions-with-inventory-status "Active"

    -- select resources to uninstall from by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "my-tag"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"

on success
    start "bar"
end
```

Table 1117. Uninstall Component Properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component	Component that is used by the step; a step can affect a single component. All components that are associated with the application are available. To uninstall more than one component, add another uninstall step to the process.
Remove Versions with	Restricts the component versions that are affected by the step. Only versions with the selected status are affected. By default, the status `Active` refers to versions that are currently deployed.

Table 1117. Uninstall Component Properties (continued)

Field	Description
Status	
Component Process	The component process to run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Uninstall Type	The **All Existing** option removes all the versions of this component that are in the inventory. The **All Selected For Process** option removes only the versions that you specify when you start the application process.
Maximum number of concurrent processes	The maximum number of concurrent processes to run. This setting limits the number of processes that run simultaneously to uninstall component instances. For example, if you set the maximum number of concurrent processes to 2, only two instances of the component are uninstalled at a time, even if many instances of the component are currently installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.

Example

Example

```
start is

    start A

    start B

end



uninstall-component step A is

    component "component-a"

    process "process-a"

    select-versions all

    select-versions-with-inventory-status "status-a"

    select-resources-by-tag "tag-a"

    fail-fast true

    ignore-child-warnings true

    max-concurrent-processes 100

    precondition-script "script-a"

    property "prop-a1" = "prop-a1-value"

    property "prop-a2" = "prop-a2-value"

    property "prop-a3" = "prop-a3-value"

on success

    finish

end



uninstall-component step B is

    component "component-b"

    process "process-b"

    select-versions selected-with-process

    select-versions-with-inventory-status "status-b"
```

```
on success

    finish

end
```

uninstall-multiple-components

The `uninstall-multiple-components` step starts child component processes for uninstalling multiple components. This step is only valid in an application process.

Format

```
uninstall-multiple-components step "foo" is
    -- name of the component process to run for each component; required
    process "deploy"

    -- mode for selecting versions to uninstall; required
    --   all - uninstall all versions installed
    --   selected-with-process - uninstall versions selected with the process
    select-versions all

    -- select versions for uninstall with this inventory status; required
    select-versions-with-inventory-status "Active"

    -- select components to uninstall by component tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-components-by-tag "comp-tag-a"

    -- select resources to uninstall from by resource tag; optional
    -- before 7.1.1.0, the tag must be specified by internal database ID
    select-resources-by-tag "res-tag-a"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes per component; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- maximum number of components to uninstall concurrently; optional
    -- default is 100
    max-concurrent-components 100

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """
```

```
on success
    start "bar"
end
```

Table 1118. Uninstall Multiple Components properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Component Tag	A user-defined component tag that is used to select components. Only components with the specified tag are modified by this step.
Component Process	The component process to run.
Limit to Resource Tag	The user-defined resource tag that determines which resource runs the process. Only a resource with this tag, or a resource that has a parent with this tag, runs the process. See Adding tags to objects on page 2521.
Remove Versions With Status	Restricts the components that are affected by the step. Only components with the selected status are affected. Available statuses: `Active` refers to components that are currently deployed.
Maximum number of concurr	The maximum number of components to uninstall at one time. For example, if you specify 5, this step runs the uninstall component processes for five components at a time. To run this step on all components at the same time, specify `-1`. To limit the number of components to access at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer.

Table 1118. Uninstall Multiple Components properties (continued)

Field	Description
ent com pon ents	
Maximum number of concurrent processes	The maximum number of concurrent processes to run, per component. This setting limits the number of processes that run simultaneously to uninstall each component. For example, if you set the maximum number of concurrent processes to 2, only two instances of a component are uninstalled at a time, even if many instances of the component are currently installed. To run an unlimited number of concurrent processes, specify `-1`. To limit the number of processes to run at once, as in rolling deployments, specify an integer. The server attempts to resolve the value to an integer. If the value does not resolve to an integer, then the `-1` value is used by default. You can use a property in this field, as long as the property resolves to an integer. **Note:** The maximum number of component processes that can run concurrently is limited by the hardware that hosts the agent.
Fail Fast	If this check box is selected, the step does not start more processes after one process fails.
Precondition	A JavaScript 1.7 script that defines a condition that must exist before the step can run. The condition must resolve to `true` or `false`. In the script, do not use the `${p:component.myProperty}` notation. For example, to check the value of a component property in a component process, use `properties.get("myProperty") == "myValue"`. See Property contexts on page 2508 for information about property access.
Uninstall Type	The **All Existing** option removes all the versions of this component that are in the inventory. The **All Selected For Process** option removes only the versions that you specify when you start the application process.

Example

Example

```
start is

    start A

    start B

end
```

```
uninstall-multiple-components step A is

    process "process-a"

    select-versions all

    select-versions-with-inventory-status "status-a"

    select-components-by-tag "comp-tag-a"

    select-resources-by-tag "res-tag-a"

    fail-fast true

    ignore-child-warnings true

    max-concurrent-processes 42

    max-concurrent-components 24

    precondition-script "script-a"

on success

    finish

end



uninstall-multiple-components step B is

    process "process-b"

    select-versions selected-with-process

    select-versions-with-inventory-status "status-b"

on success

    finish

end
```

Core component process steps

Lists the various core component steps that can be used in a process.

add-inventory-status

The `add-inventory-status` step adds a status to a component resource.

Format

```
add-inventory-status step "foo" is
    -- name of the status to add to the component resource; required
    status "Active"
on success
    finish
end
```

Properties

Table 1119. Add Inventory Status properties

Field	Description
Status	The component resource that the process is running against will have an inventory entry created for it with the specified status. `Active` indicates that the component version is deployed to its target resource. You can create other statuses and have those statuses applied to component versions when processes are complete. The status is displayed on the Inventory panes for the component itself and environments that ran the process.

Example

Example

```
start is
    start "A"
    start "B"
end

add-inventory-status step A is
    status Active
on success
    finish
end

add-inventory-status step B is
    status inventory-status-1
on success
    finish
end
```

component-manual-task

The `component-manual-task` step pauses the process until a manual approval is completed. This step is only valid in a component process.

Format

```
component-manual-task step "foo" is
    -- users that can approve the task; required
    -- approval restriction to the deploying user
    restrict-approval-to deploying-user

    -- name of notification template; optional
    notification-template "TaskCompleted"

    -- property-definition establishes a property definition; optional, repeatable

    -- a text property definition with all options
    property-definition p1-text is text with
        -- label for the property definition; optional
        -- common to all property defintion types
        label "p1-label"

        -- description for the property definition; optional
        -- common to all property defintion types
        description "p1-description"

        -- regex pattern for the property definition's property values; optional
        -- an empty pattern is equivalent to ".*"
        -- common to all property defintion types
        pattern "p1-pattern-.*"

        -- a value is required for property defined by this property definition; optional
        -- common to all property defintion types
        required true

        -- default value for text properties created by this property definition; optional
        default "p1-pattern-default"
    end

    -- a text property definition with no options
    property-definition p2-text-simple is text

    -- a text area property definition with all options
    property-definition p3-textarea is text-area with
        -- common options; optional
        label "p3-label"
        description "p3-description"
        pattern ".*"
        required true

        -- default value for text area properties created by this property definition; optional
        default
            """
            p3-default-line1
            p3-default-line2
            p3-default-line3
            """
    end

    -- a text area property definition with no options
    property-definition p4-textarea-simple is text-area
```

```
   -- a text area property definition with all options
   property-definition p5-secure is secure with
       -- common options; optional
       label "p5-label"
       description "p5-description"

       -- default value for secure properties created by this property definition; optional
       -- currently, default values for secure properties do not function properly and their use is not
recommended
       default "p5-pattern-default"
   end

   -- a secure property definition with no options
   property-definition p6-secure-simple is secure

   -- a checkbox property definition with all options
   property-definition p7-checkbox is checkbox with
       -- common options; optional
       label "p7-label"
       description "p7-description"
       pattern "true|false"
       required true

       -- default value for checkbox properties created by this property definition; optional
       -- permitted values are 'true' or 'false'
       default true
   end

   -- a checkbox property definition with no options
   property-definition p8-checkbox-simple is checkbox

   -- a date-time property definition with all options
   property-definition p9-datetime is date-time with
       -- common options; optional
       label "p9-label"
       description "p9-description"
       pattern ""
       required true

       -- default value for date-time properties created by this property definition; optional
       -- time is specified as milliseconds from 1970-01-01 00:00:00 UTC.
       default millis 1594814965125
   end

   -- a date-time property with an xml schema format default value
   property-definition p10-datetime-xml is date-time with
       -- default value for date-time properties created by this property definition; optional
       -- time is format is specified at https://www.w3.org/TR/xmlschema-2/#dateTime
       default xml "2000-01-20T12:00:00+12:00"
   end

   -- a date-time property definition with no options
   property-definition p11-datetime-simple is date-time

   -- a select property definition with all options
   -- at least one value must be supplied
```

```
    -- at most one value may be selected as the default value
    property-definition p12-select is select of
        -- a possible value in for the property; optional, repeatable
        value "p12-value1"

        -- a value with an optional label
        value "p12-value2" as "Value 2"

        -- a value with a label and optionally selected as the default value
        value "p12-value3" as "Value 3" selected
    with
        -- common otptions; optional
        label "p12-label"
        description "p12-description"
        pattern "p12-value.*"
        required true
    end

    -- a select property definition with an unlabeled value selected as the default and no common options
    property-definition p13-select-default-unlabeled is select of
        value "p13-value1" selected
        value "p13-value2"
        value "p13-value3"
    end

    -- a select property definition with no options
    property-definition p14-select-simple is select of
        value p14-value1
        value p14-value2
        value p14-value3
    end

    -- a multi-select property definition with all options
    -- a date-time property definition with all options
    -- one or more values may be selected as the default value
    property-definition p15-multiselect is multi-select of
        -- a possible value in for the property; optional, repeatable
        value "p15-value1"

        -- a value with a label and optionally selected as one of the default values
        value "p15-value2" as "Value 2" selected

        -- a value with a label and optionally selected as a second default value
        value "p15-value3" as "Value 3" selected
    with
        -- common options; optional
        label "p15-label"
        description "p15-description"
        pattern "p15-value.*"
        required true
    end

    -- a select property definition with an unlabeled values selected as the defaults and no common options
    property-definition p16-multiselect-default-unlabeled is multi-select of
        value "p16-value1" selected
        value "p16-value2" selected
        value "p16-value3"
```

```
    end

    -- a select property definition with a single unlabeled value selected as the default and no common options
    property-definition p17-multiselect-one-default is multi-select of
        value "p17-value1" selected
        value "p17-value2"
        value "p17-value3"
    end

    -- a multi-select property definition with no options
    property-definition p18-multiselect-simple is multi-select of
        value "p18-value1"
        value "p18-value2"
        value "p18-value3"
    end
on success
    finish
end

component-manual-task step "bar" is
    -- approval restricted to certain application roles
    -- if present, at least one role must be supplied
    restrict-approval-to application-roles of
        -- role name on the standard resource type; optional, repeatable
        -- before 7.1.1.0, the role must be specified with a database ID
        role "role1"

        -- role on a custom resource type; optional, repeatable
        -- before 7.1.1.0, the resource type must be specified with a database ID
        role "role2" for "resourcetype1"
    end
end

component-manual-task step "baz" is
    -- approval restricted to certain environment roles
    -- if present, at least one role must be supplied
    restrict-approval-to environment-roles of
        role "role3"
        role "role4" for "resourcetype2"
    end
end

component-manual-task step "quux" is
    -- approval restricted to certain component roles
    -- if present, at least one role must be supplied
    restrict-approval-to component-roles of
        role "role5"
        role "role6" for "resourcetype3"
    end
end

component-manual-task step "quuux" is
    -- approval restricted to any user
    restrict-approval-to any-user
end
```

Example

Example

```
start is

    start A

    start B

    start C

    start D

    start E

end



component-manual-task step A is

    restrict-approval-to application-roles of

        role "role1"

        role "role2" for "resourcetype1"

    end



    notification-template template1



    property-definition p1-text is text with

        label "p1-label"

        description "p1-description"

        pattern "p1-pattern-.*"

        required true

        default "p1-pattern-default"

    end



    property-definition p2-text-simple is text



    property-definition p3-textarea is text-area with
```

```
    label "p3-label"
    description "p3-description"
    -- see p19-textarea-pattern for pattern
    required true
    default "p3-default-line1\np3-default-line2\np3-default-line3"
end

property-definition p4-textarea-simple is text-area

property-definition p5-secure is secure with
    label "p5-label"
    description "p5-description"
    required true
    default "p5-default"
end

property-definition p6-secure-simple is secure

property-definition p7-checkbox is checkbox with
    label "p7-label"
    description "p7-description"
    pattern "true|false"
    required true
    default true
end

property-definition p8-checkbox-simple is checkbox
```

```
property-definition p9-datetime is date-time with
    label "p9-label"
    description "p9-description"
    pattern ""
    required true
    default millis 1594814965125
end

property-definition p10-datetime-xml is date-time with
    default xml "2000-01-20T12:00:00+12:00"
end

property-definition p11-datetime-simple is date-time

property-definition p12-select is select of
    value p12-value1
    value p12-value2 as p12-value2-label selected
    value p12-value3 as p12-value3-label
with
    label "p12-label"
    description "p12-description"
    pattern "p12-value.*"
    required true
end

property-definition p13-select-default-unlabeled is select of
    value p13-value1 selected
```

```
    value p13-value2 as p13-value2-label

    value p13-value3 as p13-value3-label

end



property-definition p14-select-simple is select of

    value p14-value1

    value p14-value2

    value p14-value3

end



property-definition p15-multiselect is multi-select of

    value p15-value1 selected

    value p15-value2 as p15-value2-label selected

    value p15-value3 as p15-value3-label

with

    label "p15-label"

    description "p15-description"

    pattern "p15-value.*"

    required true

end



property-definition p16-multiselect-default-unlabeled is multi-select of

    value p16-value1 selected

    value p16-value2 selected

    value p16-value3

end
```

```
    property-definition p17-multiselect-one-default is multi-select of

        value p17-value1 selected

        value p17-value2

        value p17-value3

    end



    property-definition p18-multiselect-simple is multi-select of

        value p18-value1

        value p18-value2

        value p18-value3

    end



    -- pattern is skipped in p13-textarea above because it

    -- does not work with values containing line terminators,

    -- consistent com.urbancode.air.property.prop_def.PropDef

    property-definition p19-textarea-pattern is text-area with

        pattern "p19-textarea-pattern-pattern-.*"

        default "p19-textarea-pattern-pattern-default"

    end

on success

    finish

end



component-manual-task step B is

    restrict-approval-to environment-roles of

        role "role3"

        role "role4" for "resourcetype2"

    end
```

```
on success
    finish
end


component-manual-task step C is
    restrict-approval-to deploying-user
on success
    finish
end


component-manual-task step D is
    restrict-approval-to any-user
on success
    finish
end


component-manual-task step E is
    restrict-approval-to component-roles of
        role "role5"
        role "role6" for "resourcetype3"
    end
on success
    finish
end
```

component-run-component-process

The `component-run-component-process` step executes a component process and waits for it to complete. This step is only valid in a component process.

Format

```
component-run-component-process step "foo" is

    -- name of the component process to run; required
    process "deploy-my-component"

    -- set a component process property value; optional, repeatable
    property "component-process-property-1" = "cpp1-value"
    property "component-process-property-2" = "cpp1-value"
on success
    finish
end
```

Example

Example

```
start is

    start A

    start B

end



component-run-component-process step A is

    process "proc-1"

    property "prop-1" = "prop-value-1"

    property "prop-2" = "prop-value-2"

    property "prop-3" = "prop-value-3"

on success

    finish

end



component-run-component-process step B is

    process "proc-2"

on success

    finish

end
```

remove-inventory-status step

The remove-inventory-status step removes a status from a component resource.

Format

```
remove-inventory-status step "foo" is
    -- name of the status to remove from the component resource; required
    status "Active"
on success
    finish
end
```

Properties

Table 1120. Remove Inventory Status properties

Property	Description
Status	The component resource that the process is running against will have the specified inventory entry removed from it.

Example

Example

```
start is
    start "A"
    start "B"
end

remove-inventory-status step A is
    status Active
on success
    finish
end

remove-inventory-status step B is
    status inventory-status-1
on success
    finish
end
```

Generic process steps

Lists the various generic steps that can be used in a process.

add-process-warning

The `add-process-warning` step adds a warning message to the executing process

Format

```
add-process-warning step "foo" is
    -- a string in the body defines the warning; required
    "Warning: process partially successful"
on success
    finish
end
```

Example

Example

```
start is

    start A

end



add-process-warning step A is

    "warning"

on success

    finish

end
```

acquire-lock

The `acquire-lock` step allows acquisition of a process lock.

Format

```
acquire-lock step "foo" is
    -- define the name of the lock; required
    lock "${p:component.name}-${p:componentProcess.name}-${p:resource.name}"
on success
    start "bar"
end
```

Properties

Table 1121. Acquire Lock properties

Field	Description
Lock Name	A string value that identifies the lock.

Table 1121. Acquire Lock properties (continued)

Field	Description
	Note: Lock Name character limit is 4000.
Fail if Lock Exists	Select this checkbox to fail the step immediately if a lock with the specified name has already been created.

Example

Example

```
start is
    start "A"
end

acquire-lock step "A" is
    lock "lock-name"
on success
    finish
end
```

generic-manual-task

The `generic-manual-task` step pauses the process until a manual approval is completed. This step is only valid in a generic process.

Format

```
generic-manual-task step "foo" is
    -- users that can approve the task; required
    -- approval restriction to the deploying user
    restrict-approval-to deploying-user

    -- name of notification template; optional
    notification-template "TaskCompleted"

    -- require a user comment when approving the task; optional
    -- default is false
    comment-required true

    -- the prompt for a user comment when approving the task; optional
    comment-prompt "Enter reason"

    -- property-definition establishes a property definition; optional, repeatable

    -- a text property definition with all options
    property-definition p1-text is text with
        -- label for the property definition; optional
        -- common to all property defintion types
        label "p1-label"

        -- description for the property definition; optional
```

```
        -- common to all property defintion types
        description "p1-description"

        -- regex pattern for the property definition's property values; optional
        -- an empty pattern is equivalent to ".*"
        -- common to all property defintion types
        pattern "p1-pattern-.*"

        -- a value is required for property defined by this property definition; optional
        -- common to all property defintion types
        required true

        -- default value for text properties created by this property definition; optional
        default "p1-pattern-default"
    end

    -- a text property definition with no options
    property-definition p2-text-simple is text

    -- a text area property definition with all options
    property-definition p3-textarea is text-area with
        -- common options; optional
        label "p3-label"
        description "p3-description"
        pattern ".*"
        required true

        -- default value for text area properties created by this property definition; optional
        default
            """
            p3-default-line1
            p3-default-line2
            p3-default-line3
            """
    end

    -- a text area property definition with no options
    property-definition p4-textarea-simple is text-area

    -- a text area property definition with all options
    property-definition p5-secure is secure with
        -- common options; optional
        label "p5-label"
        description "p5-description"
        pattern "p5-pattern-.*" -- currently, patterns for secure properties do not function properly and their
use is not recommended
        required true

        -- default value for secure properties created by this property definition; optional
        -- currently, default values for secure properties do not function properly and their use is not
recommended
        default "p5-pattern-default"
    end

    -- a secure property definition with no options
    property-definition p6-secure-simple is secure
```

```
    -- a checkbox property definition with all options
    property-definition p7-checkbox is checkbox with
        -- common options; optional
        label "p7-label"
        description "p7-description"
        pattern "true|false"
        required true

        -- default value for checkbox properties created by this property definition; optional
        -- permitted values are 'true' or 'false'
        default true
    end

    -- a checkbox property definition with no options
    property-definition p8-checkbox-simple is checkbox

    -- a date-time property definition with all options
    property-definition p9-datetime is date-time with
        -- common options; optional
        label "p9-label"
        description "p9-description"
        pattern ""
        required true

        -- default value for date-time properties created by this property definition; optional
        -- time is specified as milliseconds from 1970-01-01 00:00:00 UTC.
        default millis 1594814965125
    end

    -- a date-time property with an xml schema format default value
    property-definition p10-datetime-xml is date-time with
        -- default value for date-time properties created by this property definition; optional
        -- time is format is specified at https://www.w3.org/TR/xmlschema-2/#dateTime
        default xml "2000-01-20T12:00:00+12:00"
    end

    -- a date-time property definition with no options
    property-definition p11-datetime-simple is date-time

    -- a select property definition with all options
    -- at least one value must be supplied
    -- at most one value may be selected as the default value
    property-definition p12-select is select of
        -- a possible value in for the property; optional, repeatable
        value "p12-value1"

        -- a value with an optional label
        value "p12-value2" as "Value 2"

        -- a value with a label and optionally selected as the default value
        value "p12-value3" as "Value 3" selected
    with
        -- common otptions; optional
        label "p12-label"
        description "p12-description"
        pattern "p12-value.*"
        required true
```

```
    end

    -- a select property definition with an unlabeled value selected as the default and no common options
    property-definition p13-select-default-unlabeled is select of
        value "p13-value1" selected
        value "p13-value2"
        value "p13-value3"
    end

    -- a select property definition with no options
    property-definition p14-select-simple is select of
        value p14-value1
        value p14-value2
        value p14-value3
    end

    -- a multi-select property definition with all options
    -- a date-time property definition with all options
    -- one or more values may be selected as the default value
    property-definition p15-multiselect is multi-select of
        -- a possible value in for the property; optional, repeatable
        value "p15-value1"

        -- a value with a label and optionally selected as one of the default values
        value "p15-value2" as "Value 2" selected

        -- a value with a label and optionally selected as a second default value
        value "p15-value3" as "Value 3" selected
    with
        -- common options; optional
        label "p15-label"
        description "p15-description"
        pattern "p15-value.*"
        required true
    end

    -- a select property definition with an unlabeled values selected as the defaults and no common options
    property-definition p16-multiselect-default-unlabeled is multi-select of
        value "p16-value1" selected
        value "p16-value2" selected
        value "p16-value3"
    end

    -- a select property definition with a single unlabeled value selected as the default and no common options
    property-definition p17-multiselect-one-default is multi-select of
        value "p17-value1" selected
        value "p17-value2"
        value "p17-value3"
    end

    -- a multi-select property definition with no options
    property-definition p18-multiselect-simple is multi-select of
        value "p18-value1"
        value "p18-value2"
        value "p18-value3"
    end
on success
```

```
    finish
end

generic-manual-task step "bar" is
    -- approval restricted to certain users and groups
    -- if present, at least one user or group must be supplied
    restrict-approval-to identities
        -- list of approving users; optional
        -- if present, at least one user must be listed
        users of
            -- a user that can approve; repeatable
            user Alice
        end

        -- list of approving groups; optional
        -- if present, at least one group must be listed
        groups of
            -- a group that can approve; repeatable
            group DevOpsTeam
        end
    end
end
```

Properties

Table 1122. Manual Application Task properties

Field	Description
Name	The name for the manual task.
Only Allow Deploying User To Complete	Restrict approval to the user who started the deployment.
Notification Template	Specify a notification template. The individual tasks map to the notification scheme used by the application (see Creating Notifications in a Notification Scheme on page 2081). If you do not want notifications created for the task, select **None**.
Properties	Optionally, add one or more properties to the task.
Require Comment	Select to require that users must add a comment about this manual task.
Comment Prompt	**Note:** This field applies only to generic process steps. Enter the text that requests users to add a comment about this manual task.

Example

Example

```
start is
    start A
    start B
    start C
    start D
end

generic-manual-task step A is
    restrict-approval-to deploying-user

    notification-template template1
    comment-required true
    comment-prompt "comment-prompt1"

    property-definition p1-text is text with
        label "p1-label"
        description "p1-description"
        pattern "p1-pattern-.*"
        required true
        default "p1-pattern-default"
    end
on success
    finish
end

generic-manual-task step B is
    restrict-approval-to identities
        users of
            user user1
            user user2
        end
        groups of
           group group1
           group group2
        end
    end

    comment-prompt "comment-prompt2"
on success
    finish
end

generic-manual-task step C is
    restrict-approval-to identities
        users of
            user user3
            user user4
        end
    end

    comment-required true
on success
    finish
end

generic-manual-task step D is
```

```
    restrict-approval-to identities
        groups of
            group group3
            group group4
        end
    end
on success
    finish
end
```

plugin

The `plugin` step allows you to execute a plug-in command. Note that this step is not valid for an application process.

Format

```
plugin step "foo" is
    -- define name of plugin; required
    plugin "Shell"

    -- define name of plugin command; required
    command "Shell"

    -- absolute path of the working directory for the step; optional
    -- if undefined, the default working directory for the process is used
    working-directory "/working/directory"

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
       """
       true
       """

    -- define properties that configure the step; optional
    -- may be repeated
    property "prop1" = "value1"
    property "prop2" = "value2"
on success
    start "bar"
end
```

Properties

Refer to the plug-in documentation in the HCL official site for detailed information about the specific values required to configure the plug-in.

Example

Example

```

start is
    start A
    start B
end
```

```
plugin step A is
    plugin plugin-value
    command command-value
    property prop1 = prop1-value
    property prop2 = prop2-value
    property prop3 = prop3-value
on success
    finish
end

plugin step B is
    plugin plugin-value
    command command-value
    working-directory "/working/directory"
    precondition-script "true"
on success
    finish
end
```

run-generic-process

The `run-generic-process` step executes a generic process and waits for it to complete.

Format

```
run-generic-process step "foo" is
    -- name of the component process to run; required
    process "generic-process-1"

    -- resource tree path where process will execute; optional
    -- default is the process's default path
    resource-path "${p:resource.path}"

    -- suppress warnings generated by child; optional
    -- default is false
    ignore-child-warnings true

    -- set a process property value; optional, repeatable
    property "prop-1" = "prop-value-1"
```

Properties

Table 1123. Run Generic Process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Process	The generic process to run.
Resource Path	The resource on which to run the generic process.
Other properties	After you select a generic process, the properties for that process are listed. You can set values here to pass to the process.

Example

Example

```
start is
    start A
    start B
end

run-generic-process step A is
    process "proc-1"
    resource-path "resource-path-1"
    ignore-child-warnings true
    property "prop-1" = "prop-value-1"
    property "prop-2" = "prop-value-2"
    property "prop-3" = "prop-value-3"
on success
    finish
end

run-generic-process step B is
    process "proc-2"
on success
    finish
end
```

run-generic-process-for-each-affected-resource

The `run-generic-process-for-each-affected-resource` step executes a generic process for each resource affected by the application process and waits for it to complete. This step is only valid in an application process.

Format

```
run-generic-process-for-each-affected-resource step "foo" is
    -- name of the generic process to run for each affected resource; required
    process "generic-process-1"

    -- don't start further children after a failure; optional
    -- default is false
    fail-fast false

    -- suppress warnings generated by children; optional
    -- default is false
    ignore-child-warnings false

    -- maximum number of concurrent processes; optional
    -- default is unlimited
    max-concurrent-processes unlimited

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
        """
        true
        """

    -- set a generic process property value; optional, repeatable
```

```
    property "generic-process-property-1" = "cpp1-value"
    property "generic-process-property-2" = "cpp1-value"
on success
    finish
end
```

Properties

Table 1124. Run Generic Process properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Process	The generic process to run.
Resource Path	The resource on which to run the generic process.
Other properties	After you select a generic process, the properties for that process are listed. You can set values here to pass to the process.

Example

Example

```
start is

    start A

    start B

end



run-generic-process-for-each-affected-resource step A is

    process "proc-1"

    fail-fast true

    ignore-child-warnings true

    max-concurrent-processes 10

    precondition-script "script-1"

    property "prop-1" = "prop-value-1"

    property "prop-2" = "prop-value-2"

    property "prop-3" = "prop-value-3"

on success
```

```
    finish

end



run-generic-process-for-each-affected-resource step B is

    process "proc-2"

on success

    finish

end
```

set-final-process-status

The `set-final-process-status` step that also appears as **Set Status** in the Web UI, forces the outcome status of a process to success or failure.

Format

```
set-final-process-status step "foo" is
    -- set workflow result to 'success' or 'failure'; required
    status success
on success
    finish
end
```

Properties

Table 1125. Set Status property

Field	Description
Status	`Failure` or `Success`.

Example

Example

```
start is
    start "A"
    start "B"
end

set-final-process-status step "A" is
    status success
on success
    finish
end
```

```
set-final-process-status step "B" is
    status failure
on success
    finish
end
```

shell

The `shell` step is a shortcut for the `plugin` step that specifies the Shell plug-in and the Shell command. Note that this step is not valid in an application process.

Format

```
shell step "foo" is
    -- a string in the body defines the script; required
    """
    msg="hello world A"
    echo $msg
    """

    -- run the script without waiting for it to complete; optional
    -- "output.log" is the path of a file that captures the output
    run-as-daemon "output.log"

    -- absolute path of the working directory for the step; optional
    -- if undefined, the default working directory for the process is used
    working-directory "/working/directory"

    -- javascript precondition script that must evaluate to true for the step to execute; optional
    precondition-script
       """
       true
       """
on success
    start "bar"
end
```

Properties

Refer to the Shell plug-in documentation for detailed information about the specific properties and values required to configure them.

Example

Example

```
start is
    start A
    start B
    start C
    finish
end

shell step A is
    "echo A"
on success
```

```
    finish
end

shell step B is
    "echo B"
    run-as-daemon "daemon.log"
on success
    finish
end

shell step C is
    "echo B"
    working-directory "/working/directory"
    precondition-script "true"
on success
    finish
end
```

start

The `start` step is a pseudo-step that defines the beginning of the process. As it is not a true step, it does not use the step keyword. The only completion condition is success. So, on success is not required or permitted.

Format

```
start is
    -- list one or more actions
    start "foo"
    start "bar"
end
```

Properties

No properties required.

Example

Example

```
start is
    start A
    start B
    start C
    finish
end
```

switch

The `switch` step allows decision logic in the process. It expands a variable by name or evaluates an expression containing a 'p' expression to a value. Based on the value, one case is selected and its actions are carried out.

Format

```
switch step "foo" is
    -- a variable or ${p:expression} to evaluate
```

```
    evaluate "variable"

-- case selected if the expression evaluates to "A"
on case "A"
    -- one or more start commands
    start "step-a"

-- additional cases follow
on case "B"
    start "step-b"

-- case selected if no other expression matches; optional
-- if present, this must be the final option
on default
    start "step-default"
end
```

Properties

Table 1126. Switch properties

Field	Description
Name	A name for the step. Other process steps can refer to this step by this name.
Property Name	The property to use for the branching. You can specify the property name by itself or in the same form as you would use in other steps, such as `${p:propertyName}`. You cannot use a secure property in this field.

Example

Example

```
start is
    start A
    start B
    start C
    start D
    start E
end

switch step A is
    evaluate prop-name-a
on case case-a1
    start case-a1-step
on case case-a2
    start case-a2-step
on default
    finish
end

switch step B is
    evaluate prop-name-b
on default
    finish
end
```

```
switch step C is
    evaluate prop-name-c
on case case-c1
    finish
end

switch step D is
    evaluate prop-name-d
end

switch step E is
    evaluate "${p:prop-name-e}"
end

shell step case-a1-step is
    "echo case-a1-step"
on success
    finish
end

shell step case-a2-step is
    "echo case-a2-step"
on success
    finish
end
```

DevOps Deploy MCP server

The Model Context Protocol (MCP) server of HCL DevOps Deploy (Deploy) enables AI models (like LLMs) to utilize specialized Deploy MCP tools. By using these tools, you can run crucial deployment tasks on an MCP client simply through chat prompts.

Connecting an MCP client to DevOps Deploy MCP server

To enable an MCP client to communicate with the HCL DevOps Deploy (Deploy) MCP server, you must define Deploy server connection details within an MCP client's configuration.

You must perform the following tasks for a successful connection of the Deploy MCP server to an MCP client:

1. **Install an MCP client:** You can use any MCP client, such as VS Code, Claude, and so on, to connect to the Deploy MCP server. Each MCP client has its own specific UI, command palette, or dedicated settings file for configuration. You must refer to the independent software vendor documentation for client-specific instructions.
2. **Add Deploy MCP server properties:** Edit the client configuration file or settings to add the Deploy MCP server properties. See MCP server properties on page 2994 and Configuration file example on page 2994.
3. **Start the server:** Start the Deploy MCP server from the MCP client.

MCP server properties

Property	Description	Required Value/Format
`"type"`	Indicate the server connection protocol.	`http`
`"url"`	Specify the full Deploy MCP server URL.	`https://deployServer_hostname:deployPort/mcp`

`"Authorization"`	Provide a valid bearer access token ID or basic access token ID.
	
	You must ensure that a realm is selected for the bearer authentication in the **System Settings**. See System settings on page 2086. Microsoft Entra ID and Okta bearer tokens are supported.
		`"Authorization": "Bearer tokenID"`
		
		`"Authorization": "Basic <base64 encoding of 'PasswordIsAuthToken:tokenvalue'>"`

Configuration file example

The following JSON format illustrates how you can define the Deploy MCP server properties in your VS Code MCP configuration file (mcp.json):

```
{
 "servers": {
  "devops-deploy-local": {
   "type": "http",
   "url": "https://deployServer_hostname:deployPort/mcp",
   "headers": {
    "Authorization": "Bearer tokenID"
   }
  }
 },
 "inputs": []
}
```

After a successful connection is established, you can use the integrated MCP tools to perform various deployment tasks. See Using the Deploy MCP tools on page 2994.

Using the Deploy MCP tools

After you have successfully connected to the HCL DevOps Deploy (Deploy) MCP server, you can use the integrated Deploy MCP tools to perform tasks within Deploy.

You can access the following Deploy MCP tools from your MCP client:

Tools	Description	Input
list_applications	Lists all applications.	None

Tools	Description	Input
list_application_processes	Lists all processes defined in an application.	`applicationId`
list_application_environments	Lists the environments for a specific application.	`applicationId`
list_application_snapshots	Lists all snapshots defined in an application.	`applicationId`
get_deployment_status	Returns the application deployment status.	`deploymentId`
deploy_snapshot_to_environment	Deploys an application snapshot to an environment.	`applicationId`, `snapshotId`, `processId`, `environmentId`
schedule_deployment	Schedule a deployment for future execution.	`deploymentId`
get_deployment_failed_status	Retrieves information about the failed steps in a deployment.	`deploymentId`
get_deployment_step_output	Retrieves information on a step in a deployment.	`stepOutputTokenOpaque`, `firstLineIndex`
search_step_output_regex	Searches logs using regular expressions.	`stepOutputTokenOpaque`, `pattern`, `contextLineCount`
search_step_output_hangs	Identifies long intervals (hangs) between output lines in step execution logs by analyzing timestamp information in the log format. Currently, this tool only checks for hangs within failed steps.	

Tip: You can retrieve a complete list of all available tools directly from your MCP client. For specific instructions on how to use your MCP client to view this list, refer to the independent software vendor documentation.

The core functionality of the Deploy MCP tools is accessed through the chat prompts within the MCP client interface. By using these prompts, you can perform complex deployment tasks by using natural language. Examples of the chat prompts that you can use are as follows:

- **List** my applications.
- **List processes for the application** `helloApplication`.
- **Deploy** `helloApplication` **into** `dev_environment1`.
- **Why did deployment** `deployment_id` **fail**?

Note: When crafting your prompts, you can reference objects by their ID or name.

The AI agent interprets your chat prompts to intelligently select and run the appropriate MCP tool to fulfill the request.

Important: The results are highly dependent on the capabilities of the LLM that your MCP client uses, and the results are non-deterministic.

Deploying a snapshot to an environment with the DevOps Deploy MCP server

You can learn perform a basic deployment with the HCL DevOps Deploy (Deploy) MCP server by using chat prompts.

Before you begin

You must have completed the following tasks:

- Defined applications, processes, environments, and snapshots in your Deploy system.
- Connected an MCP client with the Deploy MCP server. See Connecting an MCP client to DevOps Deploy MCP server on page 2993.

About this task

You can use any or all of the following prompts in your MCP client to verify resources and initiate a deployment.

- Prompt: List my applications.

  A list of all your applications is displayed (for example, `App1`, `App2`, etc.).

- Prompt: List environments for `App1`.

  A list of all environments defined within the application `App1` is displayed (for example, `dev`, `test`, `prod`, etc.).

- Prompt: List the application processes for `App1`.

  A list of all application processes defined for the application Product1 is displayed (for example, `appProcess1`, `appProcess2`, etc.).

- Prompt: List snapshots for `App1`.

  A list of all snapshots defined for the application Product1 is displayed (for example, `snapshot1`, `snapshot2`, etc.).

- Prompt: Deploy `snapshot1` of `App1` into `test`.

  The output displays the deployment request ID initiated with the prompt.

Note: Depending on the capabilities of the LLM, you can use this prompt independently of the previous prompts.

- Prompt: Get the deployment status.

  The output displays the deployment status.

Troubleshooting and Support

Learn how to troubleshoot some common issues with HCL DevOps Deploy (Deploy).

For assistance with Deploy, contact support.

Important: Before you initiate a support request for Deploy, you must recreate the unexpected behavior in a standard environment. You must disable or remove all third-party products, including non-standard configurations and monitoring agents (such as Dynatrace or New Relic).

Product support covers only the configurations that are described in the official documentation. You must remove external variables to ensure that the issue can be properly isolated to the core Deploy system.

Logging

The server maintains logs for auditing and troubleshooting.

Audit log

The audit log shows information about who accessed the server and what changes each user made.

Note: After upgrading to HCL DevOps Deploy (Deploy) version 7.2.3.0 or later, you must manually enable the audit log. Starting with version 7.2.3.0, Deploy uses Log4j 2.x for logging configuration to enhance security.

Diagnostics log

The diagnostics log shows a table of information about actions on the server. This table shows which user ran an action, when that action was run and what part of the server API was used.

Server output log

The server output log shows information about all the actions that are performed on the server.

Viewing logs

When you install and run the Deploy server and it's components, the information is logged in the form of install logs, audit logs, diagnostic logs, and server output logs. By using the logs, you can troubleshoot errors.

Installation and upgrade logs

After you install the server, agent, and agent relay, a log file is added to the `var/log` of the installation folder. For example, you can find the:

- Server installation log file in the *`server_installation_directory`*`/var/log/install-<date>-<time>.log` directory.
- Agent installation log file in the *`agent_installation_directory`*`/var/log/agent-install-<date>-<time>.log` directory.
- Agent relay installation log file in the *`relay_installation_directory`*`/var/log/agentrelay-install.log` directory.

Audit logs

The audit log contains entries of user activity. An entry is created whenever a user does one of the following activities:

- Performs an activity that is granted by a security permission, such as creating a component, or creating an application. These activities include permissions granted by the server configuration security type, such as managing plug-ins, system properties, and statuses. Permissions are based on the team roles that are assigned to the user.

  Note: The audit log shows the installation, upgrade, and deletion of plug-ins as well.

- Modifies security settings, such as creating a role, or assigning a user to a role.
- Performs an authentication activity, such as logging on to the Web UI, or accessing the REST API.

To view audit logs, you must be in an administration role, specifically a role with the Web UI **Settings Tab** permission. You can perform the folowing actions on the audit logs:

- Review the audit log files by logging in as an administrator and click **Settings > Audit Log**.
- Filter the log by typing in the fields at the top of each column. For example, to view the admin user's activities, select **admin** in the **User Name** filter field.
- View only actions in which an element was created by specifying `CREATE` in the **Event Type** filter field.
- Remove old entries from the audit log, click **Cleanup Log**, specify the date and time, and click **Submit**. All log entries prior to that date and time are permanently deleted. Deleting audit logs creates a log entry of event type Audit Log Cleanup that are non-deletable.

Note: The activities of the built-in administrator user (with the user name admin) generate entries in the Audit Log page.

Diagnostics

To view the diagnostic logs, **Settings > Diagnostics**. The tabs on this page show diagnostic information:

Diagnostics Bundle

The diagnostics bundle contains information about the recent state of the Deploy server and system settings information. To download a Diagnostics Bundle zip file that contains the diagnostics information, click **Download Diagnostics Bundle** from the **Diagnostics** page.

A diagnostics-(date)-(time).zip file is downloaded to your system. The `diagnostics-(date)-(time).zip` file contains the following folders:

Directory	Contents
`bin`	The classpath.conf and set_env files from the server's `bin` directory.
`conf`	Various configuration details from the server's `conf` directory.
`diagnostics`	JSON files related to performance of the server.
`logs`	The upgrade logs. **Note:** The upgrade logs located in the *`server_installation_directory`*`/var/log/` directory are also copied to the *`server_installation_directory`*`/var/diagnostics-(date)-(time).zip` file.
`opt`	Tomcat settings from the server's `opt` directory.
`patches`	A collection of patches applied to the server. The `patches` folder appears only if the patches are applied.

REST Call Log

This log shows all the requests that the server receives, including requests from the web interface, the REST API, and the command-line client. You can filter this table by typing values in the fields at the top and then pressing Enter.

Metadata Indexing

This page shows the state of the server indexes, which stores information about elements such as components and processes. If indexing is not complete, you might not be able to use certain filters or to open certain processes. Indexing happens after certain server upgrades.

Database

On this page, you can analyze database query performance by clicking **Test Database Latency**. The test result gives you the following information:

- Number of queries that were completed.
- Total time to complete all of the queries.
- Total number of failed queries.
- Minimum latency for any single query.
- Maximum latency for any single query.
- Average latency for all of the queries that were completed.

Note that a maximum of 50 queries are run at a time. From the **Test Results** window, use the **show as text**/**show as table** option to send the result in desired format. The result is available in the diagnostics bundle as well.

Output logs

You can get detailed logs for processes, including the all of the information and logs from each step. To troubleshoot a step, click **View Output Log** from the options column. The **Output Log** window opens and displays the output along with the timestamp. To download these logs, open the process request, and click **Download All Logs**.

To open the server output log file directly from UI, click **Settings > Output Log > Download**. In an HA cluster, logs of all servers are downloaded in a zip format.

The Deploy server log file is also located in the following directory: *`server_installation_directory`*`/var/log/deployserver.out`.

The information that is written to the log file is determined by the settings in the **Logging** tab. You can modify the configuration by clicking **Add Logger** from **Settings > System > Logging**. See Managing loggers on page 3000 for more information.

The `deployserver.out` log displays the output as below:

```
{"ts_utc":"2022-07-07T12:13:44.849Z","ts_loc":"2022-07-07T08:13:44.849-0400","lvl":"INFO","thr":"main","log":"c
om.urbancode.ds.UDeployServer","msg":"License Server URL: "}
{"ts_utc":"2022-07-07T12:13:44.884Z","ts_loc":"2022-07-07T08:13:44.884-0400","lvl":"INFO","thr":"main","log":"c
om.urbancode.ds.UDeployServer","msg":"Database User name: dbUser"}
```

Using the JQ tool, you can see the `deployserver.out` log output in the following format:

```
2022-07-21 17:31:37,274 IST INFO  main com.urbancode.ds.UDeployServer - Cleaning up temp directory
2022-07-21 17:31:37,290 IST INFO  main com.urbancode.ds.UDeployServer - Starting HA Lock Manager...
2022-07-21 17:31:37,507 IST INFO  main com.urbancode.ds.UDeployServer - Configuring Security System...
2022-07-21 17:31:38,060 IST INFO  main com.urbancode.ds.UDeployServer - Configured authentication realms in
 JAAS.
2022-07-21 17:31:42,585 IST INFO  main com.urbancode.ds.UDeployServer - done
```

For more information about the JQ tool, see https://stedolan.github.io/jq/. An example usage of JQ command in a Unix or Linux shell is:

```
jq -r '.ts_loc + " " + .lvl + " " + .thr +  " " + .log + " - " + .msg' "deployserver.out"
```

Note: The below logging exceptions are ignored by the JQ command:

```
(
{"ex_msg", "ex_trc"}
)
```

Managing loggers

You can add loggers to the root log and set their logging level to keep track of errors and some activities.

About this task

The server stores its log by using Apache Log4j version 2.x (Log4j2). The Log4j2 logging settings are stored in the `devops-deploy/server/appdata/conf/server/log4j2.xml` file, where `devops-deploy` is your Deploy installation directory. You can edit the Log4j2 settings by editing the `log4j2.xml` file or from the web UI to change the format of the log and the information that is added to the logs. The `log4j2.xml` file contains a combination of loggers that prints information of specific loggers into the server's main or root log file, that is, `deployserver.out`.

Note: After upgrading to HCL DevOps Deploy (Deploy) version 7.2.3.0 or later, you must manually enable the audit log. Starting with version 7.2.3.0, Deploy uses Log4j 2.x for logging configuration to enhance security.

For more information about customizing the Log4j2 output, see Log4j2 Configuration.

You can edit the agent's log settings using the `agent/conf/log4j2.xml` file and the agent relay's log settings using the `agent-relay/conf/log4j2.xml` file.

To manage the server loggers from web UI:

1. Open the Logging page by clicking **Settings > System > Logging**.
   Loggers are listed in hierarchical order. The first one is the root logger, which is indicated by an `*` in the UI. It typically has no name in the logging library. The other default loggers and log classes are as follows:

   ```
   com.sun.jersey.server.impl.application.WebApplicationImpl
   com.sun.jersey.spi.container.ContainerResponse
   com.urbancode.air.diagnostics
   com.urbancode.air.vc
   com.urbancode.container.tomcat
   com.urbancode.ds
   com.urbancode.encryption.keystore
   net.schmizz
   org.apache.tomcat
   org.hibernate.cache.ReadWriteCache
   org.hibernate.engine.jdbc.spi.SqlExceptionHelper
   org.hibernate.orm.deprecation
   com.urbancode.ds.cleanup
   ```

   Each logger has a level drop-down that allows you to change the logging level. The following log level types are available:
   - ALL
   - TRACE
   - DEBUG
   - INFO
   - WARN
   - ERROR
   - FATAL
   - OFF
2. To edit multiple loggers simultaneously, select the loggers from the selection column, click **Bulk Actions**, set the level as required, and save your changes.

3. To add a logger, click **Add Logger**, specify the logger name, it's logging level, and confirm your changes.
4. To delete a custom logger use the actions column or to delete multiple loggers, use **Bulk Actions**. The root and default loggers cannot be deleted. Removing a logger reverts the logger to the logging level defined by its superior in the hierarchy.

Configuring log rotation settings

You can configure the rotation and increase the size of the main log file.

About this task

Logs include time frames for when problems occur or activities happens on the HCL DevOps Deploy (Deploy) server, agent, and relays. Because log files hold a limited amount of data, data related to a problems might be overwritten. To avoid this scenario, you can increase the size of the log file or the number of rotating files.

To configure log rotation settings:

1. Go to the **Configuration** page.
2. Change one or both of the following settings:
   - **Files**: The number of total logging files, which includes the active log file and the rotated log files. The default value is 10. Note that the value cannot be less than 1.
   - **File Size**: The maximum size (in megabytes) of a log file. Once this size is reached, the active log file is rotated. The default value is 50. Note that the value cannot be less than 1.
3. Click **Save**.

Results

When you increase the number of log files, more disk space is required. A Deploy support specialist can help you determine the appropriate value to specify.

Managing the logs

With appropriate permissions, you can add, edit, enable, and delete the default logs.

About this task

The **Other Logs** page lists some default logs, such as audit, cleanup, license, login audit, and user audit. Each of these logs contains information for different purposes.

You can manage the default logs on this page with several actions.

1. Click **Settings > Logging > Other Logs**.
2. Edit a log from the actions cloumn.
   The **Edit Log** window allows you to edit the following information:
   - The rotation type of the log file, such as hourly, daily, weekly, monthly or with respect to size
   - The number of files to generate
   - The maximum size of log files

    - The enablement status of the log
    - Add or delete loggers
3. Add a log by clicking **New Log** and provide information as mentioned in step 2.

   The **New Log** feature allows you to send log data to a specific file. It is the Log4j 2.x implementation of Log4j 1.x appenders as used in v7.2.2 and earlier versions.
4. Delete a custom log.
   Note that you cannot delete default logs.
5. Enable or disable a log.
   The cleanup log is enabled by default and cannot be disabled. However, some logs are disabled by default, such as server audit and licensing.
6. **Optional:** Perform a bulk operation by selecting one or more logs, and enable, disable, or delete them simultaneously.

Determining the Java heap memory size

The Java heap is memory that contains objects used by Java programs. It contains objects that are currently in use, objects that are no longer needed, and free memory. Objects that are no longer needed are freed by the garbage collection process. The JVM heap size determines how often to run garbage collection and how long the garbage collection runs.

About this task

Tuning the heap size minimizes the amount of garbage collection processing. Insufficient Java heap space can result in out of memory errors or slow performance.

The JVM settings are specified in the `set_env` file which is in the `devops-deploy/server/bin` directory. The **JAVA_OPTS** environment variable contains the following arguments related to heap size:

-Xms

Initial Java heap size.

-Xmx

Maximum Java heap size

For the Agent add **-Xmx** as the first line in the agent's `conf/plugin-javaopts.conf` file.

Troubleshooting the DevOps Deploy server and agents

Troubleshooting information that is specific to the server and agents is provided here.

Troubleshooting security and permissions

To troubleshoot permissions, look carefully at the roles and teams of the user and the teams of the object that the user is trying to access.

User cannot access a page on the server

If a user cannot see pages on the server, make sure that the user's role has the permission to see that page. Open the user's role, go to the Web UI category, and specify the pages that the user can access. See Creating roles and assigning permissions on page 2188.

User cannot access an object

If a user cannot access an object, such as an application or environment, check the following things:

- Ensure that the object is part of a team and that the user is a member of the same team. You can add an object to a team only if you are a member of that team.
- Ensure that the user has a role on that team, and that the role includes permission to access the object.
- If the object has a security type other than the standard security type for that object, make sure that the role includes permissions for that security type.
- Ensure that the user has access to the related pages on the server, such as the **Applications** tab or **Components** tab.
- If the user can create an object but cannot set configuration options for it, make sure that the user's role has the Edit Basic Settings and Manage Teams permissions for that type, because otherwise the options for creating the object are limited.

Troubleshooting agent connections

You can answer a number of questions to help isolate the issue behind problems with agents.

By default, the logs for the agent are in the folder *agent_install*`/var/log`.

Starting with V7.0.3, you can download and display agent logs from the user interface. This feature is only available for web agents V7.0.3 and later. The **Request Agent Logs** option is displayed on the selected agent page. After clicking **Request Agent Logs**, the *agent.out* (default file name) appears under the **Available Logs** section on the same page.

If you are having trouble with an agent, see the following troubleshooting information for the state of the agent:

- A new agent isn't displayed in the user interface, or the agent is reported as `offline` on page 3004.
- The agent is reported as `connecting` on page 3006.
- The agent is reported as `online` on page 3006.
- The agent fails to connect to the server and returns errors on page 3006.

If the agent isn't displayed in the user interface, or if the agent is reported as `offline`, complete these actions:

- Verify that the agent started successfully:
  - Look for error messages in the log. If the agent started successfully, the `agent.out` file has a message that says something like this:

    ```
    2017-07-18 08:37:47,575 INFO  AgentWorkerThread com.urbancode.air.agent.AgentWorker - Agent
     started
    ```

  - Check the prerequisites for the agent. See Installing agents.

- If the agent prompt returns right away or the window flashes and closes after you try to start it, check the Java version and Java home. This situation typically happens when Java cannot be found.
- Verify that the system that hosts the agent has enough free disk space, and that the file system and permissions are properly configured.
- If the agent is configured as a service on Linux, start the agent from the command line. If you cannot start the agent from the command line, the problem is likely related to the way that the service was configured. See Running agents as services on Linux.
- Verify that the agent installation properties are correct. See Agent installation properties.
- If the agent is configured as a service on Windows, complete these actions:
    1. Remove the service and add it again. See Removing and reinstalling agents as Windows services on page 2109.
    2. Remove the service and try to start the agent from the command line. If it starts, check permissions for the user who is running the service. See step 3 of Installing agents from the command line.

- If the agent started successfully but the server shows the agent as offline or does not show the agent at all, check these things:
    - If the agent connects through an agent relay, verify that the relay is running.
    - Verify that the system that hosts the server has enough free disk space, and that the file system and permissions are properly configured.
    - Search the user interface for the agent endpoint ID from the `installed.properties` file. If an agent with the same name is already in the system, Deploy adds the agent to the user interface as `agent-<agent id>`. It is possible that the agent is there, but not by a name that you might be looking for. This can happen if you clone a virtual machine or create a cloud environment without changing the agent name.
    - If the agent is using failover, make sure that failover is configured correctly.
    - Check for licensing issues.
    - Check the network connection between the agent and the server or between the agent and agent relay:
        - Verify that the Web relays are connected to Web agents.
        - Verify that a firewall is not preventing the agent from connecting to the server or agent relay. Agents always initiate connections to servers and agent relays, not the other way around.
        - Check to see that the agent system can ping the Deploy server or agent relay.
        - If agents are not connecting correctly on high-availability environments, make sure that the servers have cluster connections to each other.
        - Firewalls might prevent the agent from communicating with the server, particularly if the agent is on a public cloud and trying to connect to an Deploy server that is within a firewall. In this case, use an agent relay.
        - If the agent is on a cloud environment and the target cloud environment can connect to the Deploy server, log in to the cloud image and check the `cloud-init` log files in this folder:`/var/lib/cloud/instance/`.
        - If an agent on a cloud environment times out with a message like `Agent did not come online after 6.0 minutes! Process not executed.`, check to see that the agent package was downloaded correctly.

If it did, try lengthening the timeout by increasing the value of the **agent_timeout** parameter in the blueprint or configuration file.
- Check for DNS issues that might prevent the agent from resolving the host name of the server or agent relay.

◦ Can the built-in admin ID see the agent online? If so, permissions might be incorrectly configured on the agent. Remember that the built-in admin ID bypasses the team's checking.

If you do not know the location of the agent that cannot connect to the server, go to the agent properties, which include the last known host name and IP address of the agent.

If the agent is reported as `connecting`, complete these actions:

1. When this situation occurs, the problem is an HTTP communication issue. To resolve this issue:
   a. Check the external agent URL in settings. In the product, click **Settings**, and then click **System Settings**. Make sure that the host name or IP address matches or resolves to the entry in the agents `installed.property` file.
   b. Use Telnet to connect to the HTTP port on the Server.

If the agent is reported as `online` but you continue to have problems, complete these actions:

1. Verify that the agent and agent relay are running the recommended and minimum versions as indicated in the user interface.
2. Check the server, agent relay, and agent logs for errors. Look for matching time stamps among the logs.
3. Enable more detailed logging by adding the following logger, and set the looging level as TRACE:

   ```
   com.urbancode.air
   ```

   Check the output for errors.

If the agent fails to connect to the server to an agent, check if the `agent.log` shows:

- **A clock skew exception** - Set the system time on the server and the computers that agents are running on to times that are the same or within a few minutes of each other. The server and agent clock times must be close together if using end-to-end encryption. They do not need to be in the same time zone, but must agree about the global time within approximately 5 minutes.
- **The SSL handshake error** - When a Web agent fails to connect to the server and returns an error `com.urbancode.ds.subsys.deploy.agent.comm.netty.server.ServerPublicKeyPinHandler - SSL handshake failed`. Configure the load balancer to use the SSL connection instead of the load balancer. In this configuration, you use TCP mode over HTTP mode in both, the front-end and back-end configurations. For instance, if the LB is HAProxy, it should treat the connection as a stream of information to proxy to a server, rather than use its functions available for HTTP requests. The following example is a front-end configuration on HAProxy:

  ```
  frontend localhost
      bind *:80bind *:443
      option tcplog
      mode tcp
      default_backend nodes
  ```

Troubleshooting agent upgrade from the Web UI

You can troubleshoot errors that occur upgrading agents from the Web UI by performing a few checks.

As part of the upgrade process, the Deploy server downloads and extracts the `air-agentupgrade.jar` file to a certain location to perform the agent upgrade. The handling of this file—specifically how it is copied and executed—varies depending on your Deploy version. To understand about the directory structure of a compact agent and non-compact agent, see Compact agent changes: Summary on page 2338.

Note: Your system administrator must monitor the `AGENT_HOME` directory structure during a WebUI upgrade to track the creation, modification and deletion of files and directories.

To resolve errors, perform the following checks:

- Verify that the `air-agentupgrade.jar` file is downloaded from the `<serverInstallDir>/opt/` directory to the `<agentInstallDir>/bin/` or `<agentInstallDir>/var/upgrade/` directory.

  In version 7.2.x and earlier, the `air-agentupgrade.jar` file is downloaded to the `<agentInstallDir>/bin/` directory. Starting with 7.3.x and later versions, the `air-agentupgrade.jar` file is downloaded to the `<agentInstallDir>/var/upgrade/` directory.

  Note: If the `air-agentupgrade.jar` file is missing, you must verify that it exists on the `<serverInstallDir>/opt/` directory. If the file exists, you must check for network connectivity issues or write permission errors on the agent.

- Verify that the agent unzipped the `air-agentupgrade.jar` file into a specific sub-directory if the `air-agentupgrade.jar` file is present in the download path.

  The following table lists the download and unzipped locations of the `air-agentupgrade.jar` file for each version of Deploy:

Version	Download location	Unzipped location
7.2.x and earlier	`<agentInstallDir>/bin/`	`<agentInstallDir>/bin/agent-upgrade/`
7.3.x and later	`<agentInstallDir>/var/upgrade/`	`<agentInstallDir>/var/upgrade/unpack/`

  If the `air-agentupgrade.jar` file is not created and unzipped in the specified location, then you must check for user permission issues.

  Note: If you use Security-Enhanced Linux (SELinux), it might block the agent user from creating directories. To resolve this issue, you must temporarily set SE-Linux to `permissive` mode.

- Refer to the logs for specific error codes if the upgrade fails or stalls for any other reasons.

In version 7.2.x and earlier, you must check the `<agentInstallDir>/bin/upgrade.out` log file. In version 7.3.x and later, you must check the `<agentInstallDir>/var/log/upgrade.log` file. You can view or download the upgrade log directly from the **Log** tab of the agent in the Web UI.

If the log exists, you must review it for the `Permission Denied` or `File Not Found` errors.

Troubleshooting processes

You can answer a number of questions to help isolate the issue behind problems and process failures.

Take these initial approaches to avoid problems with your processes and check these areas before you take more complicated troubleshooting steps:

- If a process is failing, look for information about the failure in the process request. To open a process request, click the application, and go to the **History** tab. Find the process in the table, and click **View Request**. Within the request, expand the steps until you find the step that failed. Failed steps have an error log that provides more information about the failure. All steps have an output log that includes information about the step progress and any logging messages that the step created. For information about viewing logs, see Viewing logs on page 2997.
- Step through each process element. Verify that you refer to properties on page 2509 by using the correct syntax.
- Verify that the steps are wired together, meaning that each step is connected to another step within a process. Then, verify that the correct step result is selected. That is, on the arrow that connects one step with the next, select the correct condition for proceeding: **Run On Success**, **Run On Failure**, or **Run on Both**.
- Verify that the process is wired in to the **Finish** step.
- Consider running each process step's equivalent function in a command-line environment to verify that each step works as planned. For example, if the step runs a shell script, try running the shell script on the command line to see whether the same problem happens. If the step starts an application server, make sure that you can start the application server on the command line by using the same commands.
- Verify that you are using the correct component processes type. For information about component process types, see Component process types on page 2423.
- Trace all the way through your process step-by-step. Steps are essentially commands to be run. See this IBM support article: MustGather: How to trace the execution of a process.
- Verify that no typographical errors are included in the process steps. Review each property and script carefully.
- If properties are a factor in the problem, you can view the properties for an application process by opening the process request and clicking the **Properties** tab. You can also view the properties that each component process step used by clicking **Input/Output Properties**. For more information about properties, see Properties on page 2500 and associated topics.
- Check the log for messages that pertain to memory size. You can generate a report of requirements for your operating systems and product version: Hardware requirements for a specific product.
- If you are using sudo commands, permissions can be a problem. Verify that the command is configured correctly. Also, verify that the user is configured in sudo and does not need to specify a password. Make sure that the directory

can be accessed by the operating system user that is running the Deploy agent, the impersonated user, and the tool that you are calling. For information about sudo commands, see Impersonation on Linux™ systems on page 2463.
- Review the step output for steps that cause problems. On a command line, run a problem line or step as the same user that Deploy uses to run the step or line. Be sure to pass the user and the group.

Other troubleshooting tips

- If your process branches, look closely at the way the process branches to ensure that the process is running in the way that you expect. See Branching and joining steps on page 2402.

What issues are you observing or experiencing?

- A process doesn't start on page 3009.
- A process fails, but no errors are logged on page 3009.
- Process failures occur after an upgrade on page 3010.
- A process runs, but it does not stop on page 3010.
- Processes fail intermittently on page 3010.
- Processes stop before completion on page 3011.
- Processes with branches do not run correctly on page 3011.
- Processes using impersonation do not run correctly on page 3011.

A process doesn't start

- Can you run the process while you use the built-in ID? Running the process that way checks for a permission issue on the resource.
- Is the agent running? For information about agents, see Agents on page 2336.
- Can any processes run on the agent? If required, create a simple “echo Hello World” shell step, and see whether you can run it on the agent. Running this shell step confirms that communication between the server and resource are established.
    - If the simple shell step works, what's different between the working step and the failing step?
- Are the components mapped? See Mapping resources and components to environments on page 2363.
- Are any errors logged in the step output? See Viewing logs on page 2997.
- Check the server log file for errors, permissions issues, or configuration issues. See Viewing logs on page 2997.

A process fails, but no errors are logged

- Confirm that the version of Deploy and the agents and agent relays are the same.
- Check the agent log file for errors that point to a configuration issue. For example, a port might be blocked or a URL is incorrect. Check the server log file for errors. For example, a port might be blocked or URL is incorrect. See Viewing logs on page 2997. See also Firewall and communication configuration.
- If you are using high-availability, check to see whether there was a failover.
- If this result happens for only this process, what might be different between this process and the other processes that succeed at running the step?

- Can the process succeed when the built-in admin ID runs it. When you run the process this way, incorrect permission configuration does not impact process execution.
- Does this problem happen with any other steps? If not, how is this step different?
- If you run a process with just this step, does it fail in the same way?

Process failures occur after an upgrade

- Does the new version support the installed Java™ Runtime Environment or Java™ Development kit? See Java
- Were there any errors in the upgrade process?
- Check the agent log file for errors that point to a configuration issue. For example, a port might be blocked or a URL is incorrect. Check the server log file for errors. For example, a port might be blocked or URL is incorrect. See Viewing logs on page 2997.
- If you use an agent relay, confirm that the agent relay was upgraded to the same version as the server.
    - Check the **Agent Relays** tab to verify that the agent relay version is supported. To open the **Agent Relays** tab, click the main **Resources** tab, and then click the **Agent Relays** tab. See the minimum recommended and minimum required versions for the listed agent relays in the upper-right corner of the tab. You might have to upgrade an agent relay or agent relays. For information about upgrading an agent relay, see Upgrading agent relays from the command line.
- Check the **Agents** tab to verify that the server supports the agent version. To open the **Agents** tab, click the main **Resources** tab, and then click the **Agents** tab. See the minimum recommended and minimum required versions for the listed agents in the upper-right corner of the tab. You might have to upgrade an agent or agents. For information about upgrading an agent, see Upgrading agents.
- Did you create this process in the upgraded version? If the process was created in a previous version, did it work correctly in the previous version? If so, were any changes made to the process after the upgrade?

A process runs, but it does not stop

- Has this process ever run to completion previously? See this IBM support article: MustGather: How to trace the execution of a process.
- Is the process wired to the **Finish** step?
- If the step that fails is a shell step, is the standard output is redirected for child processes to https://developer.ibm.com/urbancode/plugindoc/ibmucd/shell/5-562693/usage/?

Processes fail intermittently

- Does the server log contain exceptions that this issue generated?
- On which step does the process fail? Does the step output contain errors?
- At which times does the process fail? Check for network outages, maintenance intervals during that time.
- Is the Download Artifacts step failing with the error `java.io.FileNotFoundException: Artifact set not found`? Ensure that CodeStation has enough time to store the artifacts before they are requested, particularly in high-availability clusters, in which the servers share the artifacts. The solution for this issue is typically to add a Wait step with a few seconds wait time before the Download Artifacts step.

Processes stop before completion

- Check the agent and server log files for errors.
- Check the process system output.
- Consider running each process step's equivalent function in a command-line environment to verify that each step works as planned.
- Can the built-in admin ID run processes with success?
- If the steps that fail are shell steps, redirect the standard output of child processes to https://developer.ibm.com/urbancode/plugindoc/ibmucd/shell/5-562693/usage/.
- Do you have sufficient memory on the Deploy server?

  Tip: The workflow execution engine that runs the process runs on the Deploy server.

- On which screen or activity do you see no progress?
- How long did the process stop? How long do you expect this activity to take?
- Are there any agents for which this behavior does *not* happen? If so, what differences exist between the processes that work and the processes that do not.
- Did this behavior always happen? If not, what changed immediately before this problem surfaced?

Processes with branches do not run correctly

- Look closely at the way the process branches to ensure that the process is running in the way as expected. See Branching and joining steps on page 2402. See this IBM support article: MustGather: How to trace the execution of a process.

Processes using impersonation do not run correctly

- Reproduce the problem with a simple shell step, use the `#!` syntax at the top to specify the shell to use. For example:

  ```
  #!/bin/sh
  ```

  In the example, `/bin/sh` resolves to different shells depending on the operating system. For example, it resolves to `/bin/ksh` on AIX and Solaris operating systems and to `/bin/bash` on a RedHat Enterprise Linux operating system.
- Put the shell script of the step into a separate file and execute it from the Deploy interface or from a command line.
- Enable logging on the agent to verify that the command ran. To enable logging, add the following logger to the logging and set the logging level as DEBUG. Start the agent again for the change to take affect.

  ```
  com.urbancode.shell.impersonation.unix.UnixImpersonateCommand
  ```

- Add to the separate script file a call to `printenv` and execute the script from a Deploy step or from a command line. Compare the two processes for differences.
- Execute the separate script using the commands configured in the **sudoFormat** and **suFormat** agent properties described in the Impersonation on Linux and UNIX systems on page 2463 topic.
- In the agent `installed.properties` file, add the following property to verify that the command executes successfully after the impersonation.

```
com.urbancode.shell.impersonation.unix.scriptPostExecuteAction=echo
```

- Update your Java fixpack level or Java version to a higher level.

Troubleshooting plug-ins

Use the following procedures and tips to help troubleshoot plug-ins that you use with HCL DevOps Deploy (Deploy):

Installing a plug-in generates an error: `Error loading plugin: undefined` on page 3012.

Loading a plug-in generates errors and steps are not displayed in the process editor on page 3012.

Installing a plug-in generates an error: `Error loading plugin: undefined`

About this task

You attempt to install a plug-in, and you receive this message: `Error loading plugin: undefined`. This error message is displayed when the file system has insufficient space. The plug-in requires more space even though it seems as though enough space is available for installation.

Depending on your hardware, add more space to the file system, and then try to install the plug-in again.

Loading a plug-in generates errors, and steps are not displayed in the process editor

About this task

You installed a trial version of Deploy with IBM® DB2® as database. After starting the server, you cannot see any plug-ins and you are unable to load new plug-ins. Numerous errors are written to the log.

The pagesize must be specified correctly when the DB2 database is created. Deploy DB2 requires a pagesize of 32 k. See Configuring a DB2 database.

Troubleshooting DevOps Deploy when running as a container

When you install HCL DevOps Deploy (Deploy) server by using the provided Helm chart, two containers are created. The first is named `init` that performs some initialization steps, including initialization of your database. The second is `hcl-launch-prod`, which is the main server container. You can debug issues by checking the logs of each of these containers.

Checking the `init` container

- To get logging information for the `init` container, you must run the following command:

```
kubectl logs [pod-name] -c init
```

- A part of the initialization process involves connecting to the database and pulling the Deploy server version information from the database. If you are installing a new release of Deploy with a new database, the database is not initialized and does not contain a Deploy version. In this case, you might get the following error message:

```
[Error] Table DS_DATABASE_VERSION does not exist
```

After this error message, you see another message indicating the database has been initialized. It might take some time for the database to initialize.

When you connect to a previously initialized database, the version of the Deploy server used to initialize the database is displayed.

Checking the server container (`hcl-launch-prod`)

After the `init` container startup finishes, the main server container starts.

- You can view the installation and setup logs by running the following command:

```
kubectl logs [pod-name] -c hcl-launch-prod
```

- After the container has started, you can run the following command inside the container:

```
kubectl exec -it [pod-name] -- /bin/bash
```

The Deploy server is installed in the `/opt/hcl-launch/server` directory. You can navigate to the directory and explore settings and log files as you do with an on-premises Deploy.

Troubleshooting problems on z/OS®

To troubleshoot problems on agent computers that are running IBM® z/OS®, examine the agent log files.

Agent log files

Agent log files are stored in the UTF-8 format. To read agent log files, use an editor that supports the UTF-8 format. For example, use IBM® Rational® Developer for z Systems™ or the z/OS® UNIX™ Directory List Utility.

Alternately, convert the agent log files to a different format. To convert an agent log file to EBCDIC IBM®-1047 encoding, type the following text at a command line:

```
iconv -f utf8 -t ibm1047 agent.out > agent.out.1047
```

Common problems

Out of memory errors

If the agent does not start or the deployment tools do not run, and a message similar to the following message is displayed on the console or written to the *agent*`/var/log/stdout` log file, the OMVS segment is not in the user profile or the memory that is assigned to the user account is not sufficient for the OMVS segment.

```
JVMJ9VM015W Initialization error for library j9jit24(11): cannot initialize JIT.
Could not create the Java virtual machine.
<JIT: fatal error, failed to allocate 8192 Kb data cache>
```

To correct the problem, ensure that the agent user account meets the security requirements. To learn more, see Security requirements on IBM z/OS computers.

ISPF gateway is not configured correctly

The error message `The ISPF gateway is not configured correctly` is typically followed by an XML response that indicates the specific problem. Typical problems include the following issues:

- The user account does not have write access to the *ispf_log_directory*/WORKAREA directory.
- The user account does not have an ISPF profile.
- The ISPF gateway cannot allocate the data sets defined in the `ISPF.conf` file. By default, the `ISPF.conf` file is in the *agent*`/conf/toolkit/` directory.

EDC5112I Response temporarily unavailable

The error message `EDC5112I Response temporarily unavailable` is displayed when no z/OS® UNIX™ processes are available to run the ISPF client gateway. To work around this behavior, set the PROCUSERMAX parameter in the OMVS segment of the user account Resource Access Control Facility profile to the value specified in the security requirements. To learn more, see Security requirements on IBM z/OS computers.

ISPF.SYSTSPRT error: EDC5129I No such file or directory.

The name of the ISPF client gateway log directory must be shorter than 51 bytes. If a directory with longer name is used, you will see the following errors when using the TSO/ISPF command plug-in step or creating a component version.

```
*** fopen() error for
ispf_log_directory/WORKAREA/WORKAREA/UCDUSER.ID00xxxx.ISPF.SYSTSPRT error: EDC5129I No such file
 or
directory.
```

```
Bad call : cp: ispf_log_directory/WORKAREA/UCDUSER.ID00xxxx.ISPF.SYSTSPRT: EDC5129I No such file
 or
directory.
```

To correct the problem, use a log directory which is shorter than 51 bytes. You can do this by reinstalling the agent and toolkit or by modifying the directory in the following two configuration files:

- `agent_dir/conf/toolkit/ISPZXENV`
- `agent_dir/conf/toolkit/installed.properties`

EDC5113I Bad file descriptor or EDC5130I Exec format error

Error message `EDC5133I Bad file descriptor error` or `EDC5130I Exec format error` is produced when a process step runs in z/OS with BPX_SHAREAS=YES. This setting will help shell to run foreground processes in the same address space as the shell is running in. For more details see, Setting _BPX_SHAREAS and _BPX_SPAWN_SCRIPT

- Our guidance is to set BPX_SHAREAS as NO always.

Performing AI analysis on failed deployments

You can perform AI analysis on failed deployments to identify the root cause of the failures and troubleshoot them easily.

Before you begin

You must have completed the following tasks:

- Configured the AI settings. See Configuring the AI settings on page 2117.
- Ensured that the **Use AI** permission is enabled for you. See Permission reference on page 2192.

About this task

When you run a deployment process, it might fail at a step or multiple steps for some reason. You can troubleshoot the failed steps by using the output log files. By using the **Deployment Genie** feature, you can get insights such as an AI interpreted root cause analysis and action items intended to ensure a failure doesn't happen again. To enhance user experience, the AI is instructed to respond in the user's language.

Attention: You must consider the following points before you perform the AI analysis -

- The recommendations from the configured AI service may be inaccurate or contain errors. If suggested solutions for an error do not seem logical or effective, open a support case to investigate the issue with Deploy.
- HCL Support does not resolve problems related to your configured AI service as it is a third-party utility beyond our control.
- Do not apply suggestions that involve deleting files and folders, and executing SQL queries that modify the database.

1. Go to the **Log** tab of a deployment request.
   The **Log** tab displays all steps of the deployment process.
2. Click **Deployment Genie**.
   The **Log** tab displays only the steps that failed in the deployment process.

   Note: You can also use the **Deployment Genie** option from the menu options of a failed step.

   Trouble: When the Deploy server runs as a container in a Kubernetes cluster and you invoke the Deployment Genie, an alert dialog may display stating that a timeout was encountered. To resolve this timeout issue, modify the Emissary Ingress mapping resource for Deploy to change the `timeout_ms` value as follows:
   a. Enter the following command:
      ```
      kubectl edit mapping <releaseName>-hcl-launch-server-prod
      ```
   b. Set the `timeout_ms` value to `0`.
   c. Save your changes.
3. Start the AI analysis of a failed step by clicking the **Deployment Genie** menu option.
   The logs are sent to the AI service for analysis. The AI Analysis window displays the root cause and its probable solution in a human-readable format.

Note: Deployment Genie masks the hostname and IP address from the log file before sharing it with your chosen LLM providers for analysis. However, you must use the Secure Property feature to mask any additional sensitive data. See Output properties on page 2516.

4. Perform the AI analysis for the remaining failed steps as needed.

Results

You have performed the AI analysis of a failed deployment process.

Measure

Measure provides end-to-end visibility into the DevOps Loop, giving teams to track how value is created throughout the work flow. It consolidates testing, security, and performance metrics to identify bottlenecks, team issues, and opportunities for improvement while strengthening governance across the organization.

Introduction to releases and pipelines

A release represents the delivery of multiple applications through one or more phases of a life cycle. A typical release includes many deployments to test environments and a small number of highly orchestrated deployments to staging and production. Release managers can define milestones that lead up to the release, as well as a plan to deliver participating applications to each phase of the life cycle.

Releases can represent all kinds of events, from major events in the life of your company to comparatively minor events, like recurring maintenance. You assign the applications, which are business-meaningful pieces of a system or software that can be independently deployed to the release. You create deployment plans, which define the tasks that must be completed when a deployment runs, for your release.

Pipelines automate releases. Pipeline stages organize inputs and jobs as your code is built, deployed, and tested. Stages accept input from either Source control management (SCM) repositories or other external applications, such as HCL DevOps Deploy (Deploy). Stages and jobs run serially and enable flow control for your work.

Home page

Use the **Home** page to view value streams, releases, and insights defined in your environment.

Access the **Home** page and user interface using the URL provided by your administrator. To access the user interface, the administrator must authenticate your credentials. After logging into Measure, the **Home** page is displayed. From the **Home** page, you can view value streams, releases, and insights.

The Home page task bar allows you to access details and work with value streams, releases, and insights. Click **Value Streams**, **Releases**, or **Insights** to view respective home pages where you can manage value streams, govern software releases, and view real-time report information for your DevOps process.

If you have administrator privileges, there are several functions available. Click **Settings** ( ) on the task bar to perform the following tasks.

- Setup the calendar
- Perform integrations and install plugins
- View license management details
- Manage user authorization
- Manage user authentication
- Configure email server
- View your profile and create User access keys (UAKs)

The **Home page** displays all recently viewed value streams and releases with each represented in respective card views. Click the required card to work with value streams and releases. Click **View all** to access the **Value Streams** and **Releases** pages.

Administering

Configuring Measure

A default Measure administrator ID is included as part of Measure. It is used to install and complete initial configuration tasks.

The Measure administrator can perform the following tasks from the administrator user interface:

- Integrations
- User authentication
- Environment mapping
- Line of business mapping

Users who are part of the Deploy system team can perform the following tasks:

- Environment mapping
- Line of business mapping

Security considerations

To ensure user data security, specify a **logout url** when authenticating users. This ensures that the user is logged out of the system and prevents another person to access their account through the cached browser data.

Configuring an email server

Configuring an email server

Before you begin

To send email, you must define an SMTP (Simple Mail Transfer Protocol) server.

This task requires that you are an Measure administrator.

About this task

You can set up a mail server for sending notifications and email.

1. On the Measure dashboard, click **Settings > Email Server**.
2. Click **Configure email server**.
3. In the **Email server** window, in the **Mail server host** field, enter the mail server URL.
   For example, `smtp.gmail.com`.
4. In the **Port** field, enter the mail server port number.
5. **Optional:** If you want to secure your email communication, check **Secure mail server connection**.
6. In the **Mail server user name** field, enter the email user account name.
7. In the **Mail server password** field, enter the user account password.
8. In the **Mail server sender name** field, enter the mail server sender name.
9. In the **Mail server address** field, enter the mail server address name.
10. Click **Save**.

Note: To delete the email settings, click **Delete Settings**.

Configuring calendar settings

Use the **Calendar Settings** page to manage the appearance of calendar events.

About this task

This task requires that you are an Measure administrator.

Use the **Calendar Settings** page to add, remove, and set the presentation style such as color for events displayed in the calendar. You can define presentation order of events for days with multiple event types. Events types include the following.

- Release events and activities
- Financial blackouts, maintenance windows, release opportunities
- Holidays

The **Preview** portion of the page displays all the defined events.

1. To access the **User authentication settings** page, click the Settings icon ( ) at the top right of the user interface.
   Click **Calendar**.
2. To create an event, click **Add event type** for the type of event to be added.
   There is an **Add event type** for each event type. Type the name of the event and click the check mark icon.

a. To change the presentation style and color, place the cursor over the default style created and right-mouse click. Optionally, click the edit icon ( ).

b. Use the left and right arrow to set the priority for the event.
The left arrow moves the event up and the right arrow moves it down.

3. To remove an event, move the cursor to the event and click the delete icon ( ).

4. To modify an event, move the cursor to the event and click the edit icon ( ).
There are separate edit icons for changing the event name and presentation style.

Configuring security settings

The Measure security model is organization- and team-based. The organization is the base element; teams belong to organizations. Team members manage releases, events, and deployments. Administrators manage teams and assign users to them.

The team-based security model means that teams manage releases, reports, pipelines, events, and run deployments. Only team members can create or modify team-managed objects. Users can be assigned to more than one team. Users are identified by email address. The administrator manages the security system and manages teams and users.

After Measure is integrated with HCL DevOps Deploy on page 1885, Deploy teams and applications can be imported into Measure. When a team is imported, all team members are imported. Users are identified by email address. Administrators can also create users.

Roles and permissions

A number of roles are defined in the product that can be assigned to a user or group. Roles set boundaries on activities that a user or group member can perform.

Roles are assigned to users and groups within the team context. Users or groups are assigned to a team with a specific role. The role assigned to a user is only for the team that the user is a member. A user can be a member of multiple teams and have a different role on each.

The interlocking concept of teams, roles, and permissions ensures that users have the appropriate permissions to perform their work and not affect processes outside of their assigned scope. The interaction of these three concepts provides the mechanisms to create an infrastructure that is secure and flexible.

There are two types of roles, global and team. Global roles provide permissions to access all resources of the product. Team specific roles provide permissions to access team specific resources of the product. Users with team specific roles can view only the resources associated with that team.

You can customize the roles using the following procedure.

1. On the Measure dashboard, click **Settings** , and then click **Authorization > Roles**.
2. On the **Roles** page, click **Create role**.
3. In the **Name** field, enter the name for the role.
4. In the **Description** field, enter the description for the role.
5. In the **Scope** field, select the scope for the role.

   Note: The possible values for the scope of the roles are global and team.

6. In the **Permissions** field, select the required action under each resources.
7. Click **Save**.

Note: The users with **Team manager (global)** and **Product administrator** role can create roles.

You can assign members to global roles using the following procedure.

1. Click to assign members to roles.
2. In the **Members** field, enter or select the member to assign the roles.
3. Click **Save**.

You can use to edit roles and to delete roles.

The roles with label **Custom** are customized roles. The roles with label **static** are system generated roles. **Static** roles cannot be modified or deleted.

Note: Until new users are assigned to teams, they are automatically assigned to the Default team with no assigned role. Administrators grant users elevated permissions when they assign them to roles such as Lead Developer or Release Manager.

Note: When you create a custom role and provide permissions to the deployment plan and deployment template, you must also add group view permissions.

The following tables show the available roles.

Note: For each of the roles, the underlying resources and permissions are also scoped as team or global.

Table 1127. Calendar manager

Resource	Action	Description
Calendar	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete calendar settings.

Table 1128. Developer (team)

Resource	Action	Description
Deployment plan	**View**	View deployment plans.
Environment	**Run**	Run team specific deployment in pipelines.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**	View team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, run, and delete team specific tasks.
Team	**View**	View teams.
Value stream	**View**	View team specific value streams and pipelines.

Table 1129. Insights manager (global)

Resource	Action	Description
Dashboard	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete dashboards.
Integration	**View**	View plugin integrations.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Value stream	**View**	View value streams and pipelines.

Table 1130. Insights viewer

Resource	Action	Description
Dashboard	**View**	View dashboards
Report	**View**	View reports.
Value stream	**View**	View value streams and pipelines.

Table 1131. Lead developer (team)

Resource	Action	Description
Application	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete pipeline applications.
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**, **Archive**	View, create, edit, and archive deployment templates.
Environment	**Create**, **Edit**, **Delete**, **Override**, **Run**	Create, edit, delete, and override pipelines environments. Run deployment in pipeline environments.
Gate	**Edit**, **Delete**, **Approve**, **Override**	Edit, delete, approve, and override pipeline gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore releases.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete and run task.
Team	**View**	View teams.
Value stream	**View**, **Create**	View and create value streams. View pipelines.

Table 1132. Lead release manager

Resource	Action	Description
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment plan templates.
Environment	**Run**	Run deployment in team specific pipeline environments.
Gate	**Approve**, **Override**	Approve and override team specific gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore team specific releases.

Table 1132. Lead release manager (continued)

Resource	Action	Description
Role	**View**	View roles.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**, **Add member**, **Remove member**	View specific teams. Add and remove members to specific teams.
User	**View**	View users.
Value stream	**View**	View team specific value streams and pipelines.

Table 1133. Pipeline designer (team)

Resource	Action	Description
Application	**ViewCreate**, **Edit**, **Delete**	Create, edit, and delete team specific pipeline applications.
Deployment plan	**View**, **Create**, **Edit**	View, create, and edit team specific deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment templates.
Environment	**Create**, **Edit**, **Override**, **Delete**, **Run**	Create, edit, and delete team specific pipelines environments. Run deployment in team specific pipeline environments.
Gate	**Edit**, **Delete**	Edit and delete team specific pipeline gates.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete team specific tasks.
Team	**View**	View teams.
Value stream	**View**, **Create**	View and create team specific value streams. View pipelines.

Table 1134. Pipeline executor (team)

Resource	Action	Description
Environment	**Run**	Run deployment in team specific pipeline environments.
Value stream	**View**	View team specific value streams and pipelines.

Table 1135. Product administrator

Resource	Action	Description
Account setting	**View**, **Edit**	View and edit account settings.
Application	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete pipeline applications.
Authentication configuration	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete authentication configurations.
Calendar	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete calendar settings.
Data retention policy	**Add policy**, **View**, **Edit**, **Delete**, **Execute**	Add, view, edit, delete, and execute the data retention policy.
Dashboard	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete dashboards.
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore deployment plans.
Deployment template	**View**, **Create**, **Edit**, **Archive**	View, create, edit, and archive deployment templates.
Email server configuration	**View**, **Edit**	View and edit email server configurations.
Environment	**Create**, **Edit**, **Override**, **Delete**, **Run**	Create, edit, override, delete, and run pipeline environments.
External User Token	**View**, **Edit**	View and edit external tool user tokens.
Gate	**Edit**, **Delete**, **Approve**, **Override**	Edit, delete, approve, and override pipeline gates.
Group	**View**, **Create**, **Edit**, **Delete**, **Add member**, **Remove member**, **Add role**, **Remove role**	View, create, edit, and delete groups. Add and remove members to groups. Add and remove roles to groups.
Integration	**View**, **Create**, **Edit**, **Delete**, **Upgrade**	View, create, edit, delete, and upgrade plugin integrations.
Plugin	**View**, **Create**, **Edit**, **Delete**, **Install**, **Download**	View, create, edit, delete, install, and download plugins.

Table 1135. Product administrator (continued)

Resource	Action	Description
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore releases.
Report	**View**, **Create**, **Run**	View, create, and run reports.
Role	**View**, **Create**, **Edit**, **Delete**	View, create, edit, and delete roles.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run tasks.
Team	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, create, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.
Teamspace	**View**, **Create**, **Edit**, **Add member**, **Remove member**	View, create, edit teamspace. Add and remove members to teamspaces.
User	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add access key**, **Remove access key**	View, create, edit, and delete users. Add and remove roles to teams. Add and remove access key to users.
Value stream	**View**, **Create**, **Edit**, **Delete**, **Design**	View, create, edit, delete, and design value streams.

Table 1136. Release executor

Resource	Action	Description
Deployment plan	**View**	View team specific deployment plans.
Group	**View**	View groups.
Release	**View**	View team specific releases.
Task	**View**, **Run**	View and run team specific tasks.

Table 1137. Release manager (team)

Resource	Action	Description
Deployment plan	**View**, **Create**, **Edit**, **Archive**, **Restore**	View, create, edit, archive, and restore team specific deployment plans.
Deployment template	**View**, **Create**, **Edit**	View, create, and edit deployment plan templates.
Environment	**Run**	Run deployment in team specific pipelines.
Gate	**Approve**, **Override**	Approve and override team specific gates.
Group	**View**	View groups.

Table 1137. Release manager (team) (continued)

Resource	Action	Description
Integration	**View**	View plugin integrations.
Release	**View**, **Create**, **Edit**, **Lock**, **Unlock**, **Archive**, **Restore**	View, create, edit, lock, unlock, archive, and restore team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**	View teams.
Value stream	**View**	View team specific value streams and pipelines.

Table 1138. Release participant (team)

Resource	Action	Description
Deployment plan	**View**	View team specific deployment plans.
Group	**View**	View groups.
Integration	**View**	View plugin integrations.
Release	**View**	View team specific releases.
Task	**View**, **Create**, **Edit**, **Delete**, **Run**	View, create, edit, delete, and run team specific tasks.
Team	**View**	View teams.

Table 1139. Team manager (team)

Resource	Action	Description
Group	**View**	View groups.
Role	**View**	View roles.
Team	**View**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.
User	**View**	View users.

Table 1140. Team manager (global)

Resource	Action	Description
Group	**View**	View all groups.
Role	**View**	View all roles.

Table 1140. Team manager (global) (continued)

Resource	Action	Description
Team	**View**, **Create**, **Edit**, **Delete**, **Add role**, **Remove role**, **Add member**, **Remove member**	View, create, edit, and delete teams. Add and remove members to teams. Add and remove roles to teams.
User	**View**	View all users.

Table 1141. Value stream manager

Resource	Action	Description
Value stream	**View**, **Create**, **Edit**, **Design**, **Delete**	View, create, edit, design, and delete team specific value streams and pipelines.

Table 1142. Value stream viewer

Resource	Action	Description
Value stream	**View**	View team specific value streams and pipelines.

Table 1143. Viewer (global)

Resource	Actions	Description
Account setting	**View**	View account settings.
Authentication configuration	**View**	View authentication settings.
Calendar	**View**	View calendar settings.
Data retention policy	**View**	View the audit logs of the data retention policy.
Dashboard	**View**	View dashboards.
Deployment plan	**View**	View deployment plans.
Email server configuration	**View**	View email server configurations.
Group	**View**	View all groups.
Integration	**View**	View configured plugin integrations.
Plugin	**View**	View available plugins.
Release	**View**	View releases.
Report	**View**	View reports.
Role	**View**	View all user roles.
Team	**View**	View all teams.

Table 1143. Viewer (global) (continued)

Resource	Actions	Description
Teamspace	**View**	View all teamspaces.
User	**View**	View all users.
Value stream	**View**	View deployment plans and value streams. Create user access token.

Table 1144. Viewer (team)

Resource	Action	Description
Deployment plan	**View**	View deployment plans.
Group	**View**	View team specific groups.
Release	**View**	View team specific releases.
Role	**View**	View team specific user roles.
Team	**View**	View specific teams.
User	**View**	View team specific users.
Value stream	**View**	View team specific value streams and pipelines.

Role mapping limitation

In Loop, Measure uses Keycloak groups to map roles. You can refine the permissions by using the authorization features. Measure supports the following roles:

- Product administrator
- Non-admin user

These roles define the baseline access level. You can extend this baseline access by using authorization within Measure.

Role mapping matrix for Measure

Keycloak group	Loop role	Supported	Notes
User group	Product administrator	No	Users in the Keycloak user group cannot be directly assigned product administrator priviliges in Measure.
User group	Non-admin user	Yes	Standard user access is supported.

Keycloak group	Loop role	Supported	Notes
Admin group	Non-admin user	No	Admin grup users cannot be downgraded to non-admin user via Loop role mapping.
Admin group	Product administrator	Yes	Administrator access is correctly mapped from Keycloak.

Authorization behavior in Measure

You can elevate permission for users in the Keycloak non-admin user group by using the authorization feature in Measure.

When configuring these settings, note the following behavior:

- You can grant non-admin users additional access rights directly at the Measure level.
- You can manage these permissions independently of Keycloak group role mapping.
- Authorization controls what actions a user can perform within Measure, but it does not change the user's Keycloak group.
- Role mapping changes in Measure do not synchronize back to Keycloak groups.

Note:

- Keycloak mapping supports only product administrator and non-admin user roles.
- Keycloak defines baseline access, while Measure authorization extends these permissions for non-admin users.
- Role mapping does not synchronize back to Keycloak groups.

Managing users and authentications

Use **Authorization** and **Authentication** to manage users and authentication.

Users can be created, modified, or removed in Measure.

Users can be authenticated from the systems detailed in the following table:

Table 1145. User authentication methods

Authentication method	UI	API
Local on page 1812	Supported	Not supported
LDAP on page 1813	Supported	Not supported
OpenID Connect (OIDC) on page 1817	Supported	Not supported

Table 1145. User authentication methods (continued)

Authentication method	UI	API
SSO on page 1817	Supported	Not supported
User access key (UAK) on page 1820	Not supported	Supported

API authentication

UAK is the recommended method through the API. An example for UAK is:

`curl -X POST -k -H 'Content-Type: application/json' -H 'Authorization: UserAccessKey xxxxxxxxxxxxxxxxxxxx'`.

Details for configuring UAKs are found here on page 1820.

Managing users

Use the **Users** page to create, edit, or remove users.

About this task

This task requires that you are an Measure administrator.

Users can be created, edited, or removed from **Authorization** on the **Settings** page.

To create a user, complete the following steps:

1. Click **Settings > Authorization > Users > Create user** to display the **Create user** window.
2. Complete the required fields.
3. Click **Save**.

What to do next

To modify or remove a user, click **Settings > Authorization > Users** and perform the following steps.

- To modify a user:
  - Click the overflow menu ⋮ and select **Edit** for the required user.
  - Change the required fields in the **Edit user** window.
  - Click **Save**.

- To remove a user, click the overflow menu ⋮ and select **Delete** for the required user.

  Note: Users are deleted from all teams and groups. Users' access keys are also deleted. You can use a deleted user email address to create a new user. The new user is created with default permissions. The user cannot retain previous roles and permissions after deletion.

Note: Additional user authorization topics include teams, groups, and user access keys.

Authenticating users with LDAP

Use the **User authentication settings** page to manage user authentication.

Before you begin

You must have the URL of the external LDAP server that you intend to authenticate users. This task requires that you are an Measure administrator.

About this task

An LDAP realm identifies users and groups and defines rules how to search users and groups. When unknown users attempt to log on, an external LDAP server authenticates them by using the realm parameters that you configure. To configure an LDAP authentication realm, you identify the URL of the LDAP server, and define valid searches.

To create an LDAP realm, complete the following steps:

1. From the Measure dashboard, page, click **Settings > Authentication**.
2. Click **Configure LDAP** .
3. On **LDAP** page, in the **Name** field, enter a name for the realm configuration.
   The value is an arbitrary label that does not effect the other settings.
4. In the **LDAP URL** field, enter the URL for the LDAP that you use for authentication.
   Separate multiple servers by commas.
   Example
   For example, `ldap://ldap_server.my_domain.com:389,ldap://ldap_server.my_domain2.com:389`.
5. Specify whether anonymous searches are allowed by completing one of the following options:
   Choose from:
   - If the LDAP server allows anonymous searches, select **Search Anonymously**.
   - For authenticated searches, clear the **Search Anonymously** check box, and then enter the **Bind DN** and **Bind credentials**. Velocity uses these fields to authenticate users when it connects to the LDAP server. For example, `cn=velocity,ou=applications,dc=mydomain,dc=com`.
6. **Optional:** In the **Scope when searching LDAP users** area, specify a search scope by selecting one of the following options:
   Choose from:

- **Subtree**. Select this option when user entries are direct children of the **Search base**.
- **One level**. Select this option if all user entries are direct grandchildren of the **Search base**.
- **Base**. Select this if option if user entries are two or more levels below the **Search base**.

The scope is relative to the **Search base** selected in the next step. It is a good practice to make the scope as narrow as possible.

7. In the **Search base** field, enter the user search base.
   Example
   The starting directory for the search, such as `ou=employees,dc=mydomain,dc=com`.
8. In the **Search filter** field, enter the search filter.
   Example
   The LDAP filter expression that is used when searching for user entries. The user name replaces the *{{username}}* variable in the search pattern, for example, `uid={{username}}`.
   If the value is not part of the DN pattern, enclose the value in parenthesis, for example, `(mail={{username}})`. For more information, see the help information for your LDAP server and look for information about creating user search filters.
9. **Optional:** In the **Bind property** field, enter a search expression.
   Example
   This is the name of the LDAP attribute that contains the **Bind DN** specified earlier. The default value is `dn`.
10. **Optional:** In the **Name attribute** field, enter the LDAP user name.
    This is the name of the LDAP attribute that contains the user's full name. Examples are `cn` and `displayName`.
11. In the **Email attribute** field, enter the user email address.
    This is the name of the LDAP attribute that contains the user's email address. For example, `mail`.
12. In the **Role definition** area, specify a role by completing one of the following options:

    a. Select **Role in LDAP reference their members** if you want to find group membership by searching roles, and then define the **Group search filter**.

    Note: The **Group search filter** is the LDAP filter expression that is used when you search for user entries. The user name replaces the {1} variable in the search pattern and the full user distinguished name (DN) replaces the {0} variable, for example, `(&(uniqueMember={dn})(cn=BSO*))`. The user name replaces the `{username}` variable in the search pattern and the full user distinguished name replaces the `{dn}` variable.

    a. Select **User roles are defined as an attribute on that user** if you want to find group membership by using this attribute, and then define the **Group DN Attribute** and **User Group Attribute** fields.
    The **Group DN Attribute** is the name of the LDAP attribute on group entries, whose value is the group's distinguished name. For example, `dn`. The **User Group Attribute** is the name of the LDAP attribute on user entries, whose value is the distinguished name of a group of which the user is a member. For example, `memberOf`.

13. In the **Group search base** field, enter the directory that is used for group searches.
    Example
    For example, `ou=employees,dc=mydomain,dc=com`.

14. In the **Group name attribute** field, enter the name of the entry that contains the users' group names in the directory entries that are returned by the group search.
    If this entry is not specified, no group search runs, for example, `cn`.
15. On the **Search group subtree** box, subtrees (if any) are searched. If the item is not selected, the search is limited to the **Group search base**.
16. Click **Save**.
17. Click **Import users** to verify the **LDAP** configuration settings. For the valid **LDAP** configurations, user import status is successful. For the invalid **LDAP** configurations, user import fails with the error message.

Results

The first time an unknown user attempts to log on, LDAP authorization realms are searched in an attempt to identify the user. If the user is found, a corresponding user ID is created in Measure. In addition, if the user is part of an LDAP group, that group is imported too.

When new users log on to the server and use their LDAP credentials, they are listed on the `Users` page. In most cases, do not manage user passwords or remove users from the list. If an active user is removed, they are still able to log on to the server while their LDAP credentials are valid.

Copying DevOps Deploy authentication and authorization realms into Measure

Transfer authentication and authorization realm settings from DevOps Deploy to Measure.

Before you begin
You must have both the authentication and authorization realms created in Deploy before you can copy them into Measure. This task requires that you are an Deploy and Measure administrator.

1. From the Deploy home page, click **Settings > Authentication (Users)**, and then click the required authentication realm.
2. Click the arrow to the left of **Edit** to display the authentication realm fields to be copied into the Measure fields.
3. From the Measure home page, click **Settings > User Authentication**.
4. Copy information from the DevOps Deploy authentication realm fields to the Measure fields, as described in the following table.

Table 1146. Copying authentication realm fields

Deploy authentication realm field	Measure field
Name	**Name**
Description	**Description**
LDAP URL	**LDAP URL**
Search Anonymously	**Search Anonymously** Note: Match selection to Deploy field.
Search Connection DN	**Bind DN**

Deploy authentication realm field	Measure field
Search Connection Password	**Bind credentials**
User Search Base	**Search base**
User Search Filter	**Search filter** Note: Do not copy the **User Search Filter** field. In the **Search filter** field, enter `mail={{username}}`.
Name Attribute	**Name Attribute**
Email Attribute	**Email Attribute**

5. Match the **Search User Subtree** selection with the **Scope when searching LDAP users** selection.
   If **Search User Subtree** and **Subtree** are not selected, follow these guidelines for making a selection in the **Scope when searching LDAP users** field:

Options
One level indicates that only the immediate children of the entry that is specified as the search base are considered. The base entry itself is not considered, nor any descendants of the immediate children of the base entry.
Base indicates that only the entry that is specified as the search base is considered. None of its subordinates are considered.

6. From the Deploy home page, click **Settings > Authorization (Groups)**, and then click the required authorization realm.
7. Click **Edit** for the required authorization realm to display the fields to be copied into the Measure fields.
   For the following steps, you will be copying information from the Deploy authorization realm fields displayed in the **Create Authorization Realm** fields to the Measure fields.
8. Copy information from the Deploy authorization realm fields to the Velocity fields, as described in the following table.

Table 1147. Copying authorization realm fields

Deploy authorization realm field	Measure field
Group Search Base	**Group Search Base**
Group Search Filter	**Group Search Filter** Note: Do not copy the **Group Search Filter** field, enter `member={{dn}}`
Group Name	**Group Name Attribute**
Search Group Subtree	**Search Group Subtree** Note: Match selection to Deploy field.
User Group Attribute	**User Group Attribute**

9. Click **Save** on the **User authentication** page in Measure.
10. Click **Import users** to verify the **LDAP** configuration settings. For the valid **LDAP** configurations, user import status is successful. For the invalid **LDAP** configurations, user import fails with the error message.

Authenticating users with OpenID Connect (OIDC)

Use the **Authentication** page to manage user authentication with **OpenID Connect**.

Before you begin

To configure the **OIDC** authentication, you must have Measure administrator permission and privileges.

About this task

OpenID Connect is an authentication method which you can use to authenticate users.

To configure an **OIDC** authentication, complete the following steps:

1. From the Measure page, click **Settings > Authentication > OIDC**.
2. Click **Configure OIDC** .
3. On **OIDC** page, in the **Name** field, enter a name for the **OIDC** configuration.
4. In the **Description** field, enter a description for the **OIDC** configuration.
5. In the **OIDC Issuer** field, enter the issuer for **OIDC** authentication.
6. In the **Authorization URL** field, enter the URL of **OIDC** authorization provider .
7. In the **Token URL** field, enter the URL for token request with **OIDC** provider.
8. In the **User info URL** field, enter the URL for user information request with the **OIDC** provider.
9. In the **Client ID** field, enter the client ID that is registered with the **OIDC** provider.
10. In the **Client Secret** field, enter the client secret that the **OIDC** provider issues upon registration.
11. Click **Save**.
    Result

    You have successfully configured **OIDC** authentication. You can log in to Measure using **OIDC** configuration.
12. On the Measure login page, click **Login with OIDC** to log in using OIDC configuration.
    The **Authorization URL** page opens to validate the user for accessing the Measure
13. On the **Authorization URL** page, enter the username in **Username** field and password in **Password** field.
    You have successfully logged in to the Measure.
14. On the **Settings > Authentication > OIDC** configuration page, click **Delete configuration** to delete the **OIDC** configuration.

Results

You have successfully configured the **OIDC** authentication. You can log in to Measure using the **OIDC** authentication.

Creating a header-based SSO authentication integration

A header-based single sign-on (SSO) authentication realm uses an external server for authentication.

1. On the Measure dashboard, click **Settings > User authentication > SSO > Configure SSO**.
2. In the required fields, specify the following parameters:

Table 1148. SSO properties

Field	Description
Name	The name for the integration.
Full name header name	The user name on the SSO server that contains the list of users.
Email header name	The user email on the SSO server that contains the list of users. Note: For **Email header name**, there is a change to lower case in the Measure database.
Logout URL	The SSO server's logout URL.

Note: Measure supports header-based SSO and can integrate with other SSO mechanisms, such as Security assertion markup language (SAML). It is the responsibility of the user to modify the SSO scheme in use to transmit the headers to authenticate with Velocity Measure and then block the login page. The aforementioned method is different for every SSO product and mechanism.

3. Click **Save**.
   Result
   Setting up an SSO user does not automatically add the user on page 1812 to the Measure database and is explained in further detail below.

Results

When users log on to an SSO-configured Measure instance, their **Email header name** SSO credential is compared to the SSO server's user list. If the email address matches an existing user in the Measure database, a user will automatically be created in the database and listed as **SSO** in the **User Type** field.

Note: Once the SSO setup is completed properly by the admin user and that user logs out of Velocity, there is a loss of admin access privileges because of the inability to log back in as the admin. New users created via the SSO login only have the Viewer and Release participant permissions by default and the admin cannot change the permissions as stated previously. Possible solutions for this scenario include the following:

1. Set the Measure server to allow direct access from specific IPs that will allow the admin to login.
2. Create an SSO user with the Measure admin email address of `admin@admin.com`.
3. Perform an SSO login with a different browser while the local Measure admin navigates to **Settings > User authentication > Teams** and grants full admin privileges to the SSO user.

As a guideline, Item 3 is the recommended method for preserving admin access privileges to Measure.

To delete a current SSO configuration after it was created, click the **Delete Configuration** button on the **Settings** page for **SSO**.

Managing user groups

You can combine users into groups. When you grant permissions to the group, you simultaneously grant them to the group members. When you assign a group to a team, all group members are simultaneously assigned to the team.

About this task

Note: Groups cannot be imported from DevOps Deploy.

To create a group, complete the following steps:

1. On the Measure dashboard, click **Settings** , and then click **User authorization > Groups**.
2. On the Groups page, click **Create Group** and in the Add New Group window, enter a name for the group, and then click **Save**.
3. With the group selected, click **Add user**.
4. In the **User** list, select a user, and then click **Save**. The user is added to the **Members** list. You can add a user by typing the user's email address. The email address is used as the user name.

What to do next

If you configure an LDAP server, groups can be imported when new users log on for the first time.

Password management

Password management details how you can change the administrator password and retrieve the RabbitMQ password from Kubernetes on page 1820 and OpenShift on page 1820.

Changing the Administrator password

Use the **Users** page to change the administrator's password.

About this task

The admin user is the Measure super user. You set the initial admin password during installation. To change the admin password, complete the following steps.

Note: This task requires that you are an administrator.

1. Click **Settings > User authorization > Users** to access the **User authentication settings** page.
2. Click the **Change password** action for the admin user, and then enter your own password.

3. Enter the new password.
4. Click **Save**.

Retrieving the RabbitMQ™ password in Kubernetes™

Accessing the RabbitMQ password from Kubernetes.

1. From a command line window, run the following command to retrieve the data contained in the secret.

   ```
   kubectl get secret velocity-rabbitmq -o yaml
   ```

2. On the `password` line, copy the password.
   The password is Base64 encrypted.
   Result
   The screen capture for copying the password is shown in the following illustration.
3. Use a Base64 decoder to decode the password.

   a. For a Linux shell, run the following command to decode the password:
      Paste the password in the command line.
      Example

      ```
      echo -n '<password>' | base64 --decode
      ```

   b. The decoded password is displayed in the shell.

Retrieving the RabbitMQ™ password in OpenShift™

Accessing the RabbitMQ password from OpenShift.

About this task

To retrieve the RabbitMQ password in the OpenShift Container Platform application, use the following procedure.

1. Navigate to the OpenShift Web Console UI.
2. Navigate to the list of secrets (**Resources > Secrets**).
3. Navigate to the **velocity-rabbitmq**.
4. Click on **Reveal Secret** to display the value in the **password** field.

Managing User access keys (UAKs)

Create a key that you can use to authenticate API calls.

About this task

Until new users are assigned to teams, they are automatically assigned to the Default team in the Viewer and Participant roles. Users in these roles can view objects, such as releases and value streams, but they cannot create or edit them. Additionally, users in these roles can auto-generate user access tokens and access API endpoints with GET requests. Administrators grant users elevated permissions when they assign them to roles such as Lead Developer or Release Manager.

1. On the Settings page, click **My Profile**.
2. Click the **Create user access key** button.
3. In the **Name** field, enter the name for the user access key.
4. Click the **Create** button.
5. In the **User access key created** window, click to copy the **Access key** to the clipboard.
6. Save the key to a secure location.

   Note: It's important to save the key because it will not be accessible after you close the **Access key created** window.

7. Click **x** to close the **User access key created** window.

What to do next

You can use the key whenever you use plugins or external clients to integrate tools into Measure.

Integrations support the capability to auto-generate the required user access keys that is implemented on a plugin by plugin basis.

Note: Plugins are not updated at the same time and you can refer to the plugin documentation page for the latest updates. The update status of existing integrations only change when you edit the integration.

The auto-generated user access key process is detailed in the following:

1. On the Measure **Home** page, click **> Integrations > Plugins**.
2. Click the **Add integration** button for the required plugin integration to display the **Add Integration** modal window.
3. The **User Access Key** field is displayed if the plugin is not updated.

   Note: The **User Access Key** field is not displayed if the plugin is updated.

4. Fill in the required fields for the plugin.
5. Click the **Show Hidden Properties** check box to display the auto-generated properties for the integration.

   Note: In the Measure **User Access Key** field, you can enter a user access key if you have one, but it is not required.

6. Click the **Add** button.

7. Click **User Authorization > User Access Keys** to locate the plugin in the **Name** field with the format of **Integration Name** - AutoGen.

   Note: Any existing plugin integrations that are edited and saved automatically auto-generate user access keys in the aforementioned manner.

You can review the issued user access keys on the **User access key management** page, **Settings > User authorization > User access keys**. If you have administrator permissions, you can delete user access keys from that page.

Configuring log level settings

Control the amount of detail sent to log files.

About this task

Log files contain information about the data that your plugin integrations use or create. You can control the amount of information kept in the log files by setting the logging levels. You set the logging level separately for each integration.

The following list contains the logging levels:

- `ALL`
- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`
- `FATAL`
- `OFF`

Each level contains less information than the level that precedes it in the list. To keep the most verbose information, select `ALL`. The default logging level is `INFO`.

1. On the **Settings** page, click **Integrations**, and then select the **Edit** action for the integration that you want to change.

   Note: You can also set the logging level when you create an integration.

2. On the **Add Integration** page, select the logging level from the **Logging Level** list.
3. Click **Update**.

What to do next

To see a log file, use the **View log** action associated with the integration.

If you want to troubleshoot an integration, setting the logging level to `DEBUG` is a good place to start.

Configuration of data retention policy

Data retention policies in Measure help you manage the lifecycle of data objects by archiving or deleting resources based on scope, retention rules, and execution schedules.

Data Retention is an administrative governance feature available only for Velocity 5.2.5 or later, which allows you manage the lifecycle of data objects over time automatically. By establishing automated retention policies, Product Administrators can systematically archive or permanently delete versions and their linked resources, such as builds, inventory, and deployments that are no longer required for active operations.

You can create a data retention policy by providing the details, such as **Name**, **Description**, **Retention days**, and **Resource type**. You can also select the following policy **Mode**, which defines the action applied to the resource:

- Archive mode: This mode provides a temporary cleanup step. When resource types, such as versions, reach their archival time threshold, the system moves them out of your daily workspace. The data is removed from the active user interface so it stops cluttering your daily view. The data is still available in the backend and the administrator can still retrieve this data whenever required. The archived data continue to appear in reports such as Pipeline Snapshot, Deployment Audit, and Release Readiness reports.
- Delete mode: In this mode, the system removes the resource data permanently from the system's hard drives and databases. Because Measure maintains data integrity, deleting an object (such as a software version) will also cascade to delete all of its tightly coupled background data (such as its specific builds and deployment history). The deleted resources do not appear in reports such as Pipeline Snapshot, Deployment Audit, and Release Readiness reports.

Both archive and delete operations are soft and hard lifecycle controls, respectively, from a UI perspective, but neither operation preserves full recoverability through bulk restore. Bulk restore is not supported. However, archived resources can be restored manually.

If a resource is shared across multiple teams or applications, the system does not delete or archive the resource until all associated teams and applications are included in the selected scope. This process ensures that shared resources are not removed prematurely.

You can define the policy scope using the **Scope** field. The available scopes are as follows:

- Global: The policy is applied to all the resources across all users.
- Team: The policy is applied to only selected teams. You can select multiple teams for a single policy.
- Application: The policy is applied to only selected applications. You can select multiple applications for a single policy.

You can manage data retention policies from the **Settings > Policies > Data retention policy** page. Available actions include edit, delete, execute, and view logs. You can view the execution details by using **Audit Logs** to track policy runs and results.

You must take a database backup before creating the policies, because the reports, value stream, and insights metrics will be impacted. The linked resources of the Versions are as follows:

- Builds
- Deployments
- Inventory

Adding data retention policy

You can add a data retention policy to archive or delete resources in Measure based on defined scope and retention rules.

Before you begin

You must have the product administrator role. For more information about the roles, see Roles and permissions on page 1801.

1. On the Home page, click the **Settings** icon.
2. On the **Settings** page, select **Policies > Data retention policy**.
3. Click **Add policy**.
4. Enter the following details in the given fields:
    - Name: Add the name of the data retention policy.
    - Description: Add the description about the policy.
    - Retention days: Add the number of days for data retention.
5. Select **Mode** from the drop-down list:
    - Archive: The resource data is removed from the active user interface, so it stops cluttering your daily view.
    - Delete: The resource data is removed from the system.
6. Select **Resource** type, Version, from the drop-down list.

    Note: Currently, only the **Version** resource type is supported.

7. Select **Scope** from the drop-down list:
    - Global: This policy scope is applied to all resources across all users.
    - Team: This policy scope is applied to only selected teams.
    - Application: This policy scope is applied to only selected applications.
8. Click **Create policy**.

    The data retention policy is created as is in a disabled state, which is the default state. To enable the policy, follow the steps:

    a. Select the toggle button under the **Disable / Enable** field.
    b. Review the impacted policy records preview.
    c. Enter the policy name again to confirm enable, and then click **Confirm**.

    The policy is enabled and completed. The policy is scheduled to run every 24-hour interval.

Results

You have added the data retention policy.

What to do next

Click the icon to expand and view the following details about the data retention policy:

- Policy ID
- Resource
- Scope
- Mode
- Last execution detail
- Executed by
- Affected count
- Cutoff time

Managing data retention policy

You can edit, delete, run, and view policies from the policy page.

Before you begin

You must have the product administrator role with at least one data retention policy.

1. On the Home page, click the **Settings** icon.
2. On the **Settings** page, select **Policies > Data retention policy**.
   A list of available data retention policies is displayed.
3. Click the icon adjacent to your data retention policy to select any of the following options:

Options	Description
Edit	Update Name, Description, Retention days, Mode, Scope, or Resource, and click **Save policy**.
Delete	Click **Confirm** to delete the policy. **Note:** The deleted policy cannot be recovered.
Execute	Review the impacted policy records preview. Enter the policy name again to confirm execution and click **Confirm**. A product administrator can use this option to run the policy manually.
View logs	You can review the execution log and details of the data retention policy. Click **Download log file** to download the file.

Viewing the execution history of the data retention policy

You can view the execution history of data retention policies from the Audit Logs page.

Before you begin

You must have the Global viewer role. For more information about the roles, see Roles and permissions on page 1801.

1. On the Home page, click **Settings > Audit Logs**.
2. Filter or search for the policy that is executed as Data Retention category.
3. Review execution status and details.

Results

The system displays the execution details as follows:

- Name: Name of the policy
- Category: Category as DATA_RETENTION
- Operation: Mode of the policy, Archive or Delete
- Resource: Resource type is Version
- Scope: Scope of the policy, Global, Team, or Application
- Records affected: Number of records affected by this data retention policy
- Status: Status of the policy as completed or failed
- Triggered by: Who executed the policy
- Completed at: Timestamp of the data retention policy execution

Measure MCP Server - Tech Preview

A Model Context Protocol (MCP) server provides a structured way for GenAI clients to access tools, data, and actions through a consistent, secure interface. When you connect a GenAI client that supports MCP to the Measure MCP server, you can use natural-language requests to retrieve information from your Measure.

Disclaimer:

This release contains access to the MCP server feature in Velocity Measure as a Tech Preview and can be enabled by passing the `betaFeature=mcp` parameter while running the installer. The Tech Preview is intended for you to view the capabilities of MCP server offered by Velocity, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL® provides the information without obligation of support and "as is" without warranty of any kind.

The Velocity Measure MCP server exposes predefined queries that let you view Value Stream Management data, metrics, build and deployment details, workflow gate statuses, and information about teams, applications, and workspaces. This capability provides a simple, conversational way to obtain operational insights without requiring navigation of the Measure interface or creation of API requests.

The Measure MCP server offers the following tools:

- applicationById
- builds
- deployments
- latestWorkflowGateChecks
- metrics
- teamsForCurrentUser
- teamspacesForTenant
- workflow
- workflowsForTenant
- workflowStatsByIdQuery

To use the Measure MCP server, you must enable the MCP server feature flag when you run the installer. See Prerequisites to connect with the MCP server in Measure on page 1827.

You can currently connect to the Measure MCP server using the following GenAI clients:

- Claude Desktop. See Connecting to the DevOps Velocity MCP server in Claude Desktop on page 1828
- GitHub Copilot in Visual Studio Code. See Connecting to the Measure MCP server by using GitHub Copilot in Visual Studio Code on page 1829.

The MCP deployment is enabled, and the MCP pod is running in the default configuration. To reduce resource consumption, you must disable the MCP deployment when the MCP server is not in use. See Disabling the MCP Deployment in DevOps Velocity.

Prerequisites to connect with the MCP server in Measure

You can find information about the requirements and setup steps needed before connecting to the MCP server in Measure.

Before connecting to the MCP server, you must complete the following prerequisites:

1. Install **Keycloak 26.5.6**.
2. Download and run the script to create the necessary clients for the Velocity application and MCP server.

   Note: Download the compressed file from here: keycloak-setup script. Refer to the `README.md` file included in the bundle for instructions to run the script.

3. Install or upgrade the Velocity instance by providing the Keycloak URL, client secret, and JWKS URL for MCP.

   Here is a sample command to install or upgrade Velocity with the MCP configuration:

```
helm upgrade --install velocity -n <custom_namespace_name> \
                ./accelerate-<version>-helm.tgz \
                --set license=accept \
                --set url.domain= <domain_name or I.P address>\
                --set access.key=<access_key>\
                --set mongo.url=<mongodb url>\
                --set secret.tls=<tls_secrete_name>\
```

```
                --set rabbitmq.nodePort=31672 \
                --set loglevel=ALL \
                --set externalKeycloak.url=https://<keycloak_hostname>/realms/devops-velocity \
                --set
 externalKeycloak.jwksUrl=https://
<keycloak_hostname>/realms/devops-velocity/protocol/openid-connect/certs \
                --set mcp.secretValue="<mcp-velocity-client-secret>"
```

Note: `--set jwksUrl=externalKeycloak.jwksUrl=https://<keycloak_hostname>/realms/devops-velocity/protocol/openid-connect/certs` is required only for Velocity 5.2.4 and earlier versions.

Before running the Helm command, you must replace the following placeholder values with your actual configuration:

- **<keycloak_hostname>**: Replace with your Keycloak server hostname or IP address.
- **<mcp-velocity-client-secret>**: Replace with the secret value configured for the **mcp:velocity** client in Keycloak.

These parameters ensure that Velocity can authenticate users via Keycloak and connect to the MCP server securely.

For more information about installation and upgrading, refer to the following related topics:

- Installing on Kubernetes
- Installing on OpenShift
- Installing on Google Kubernetes Engine
- Upgrading Kubernetes-based installations
- Upgrading an OpenShift installation using Helm charts

Connecting to the DevOps Velocity MCP server in Claude Desktop

You can connect to the Measure Model Context Protocol (MCP) server by using the Claude Desktop application to interact with Velocity and retrieve the necessary data and insights.

Before you begin

You must have completed the following tasks:

- Installed Velocity and configured Key cloak. See Prerequisites to connect with the MCP server in Measure on page 1827.
- Configured OIDC authentication in Velocity. See Authenticating users with OpenID Connect (OIDC) on page 1817
- Obtained the MCP Server URL. For example: `https://<velocity_hostname>/mcp`
- Ensured that Claude Desktop is installed.

1. Open the **Claude Desktop** application.
2. Navigate to **Settings > Connectors**.
3. Click **Add custom connector**.
   The **Add custom connector** window is displayed.
4. Enter the name of the connector. For example: Velocity MCP server
5. Enter the Velocity MCP server URL in the **Remote MCP server URL** field.

6. Click **Add**.
   The MCP server is now added and listed on the **Connectors** page.
7. Click the **Connect** button next to the newly added Velocity MCP server entry.

   Claude Desktop redirects you to the **DevOps Loop** login page.
8. Sign in using the Velocity OIDC user credentials.

   A consent screen is displayed requesting access to MCP tools, Velocity APIs that are used internally for tool execution, and basic profile and role information.
9. Click **Yes** to approve and complete the connection.

   After successful authentication, Claude Desktop displays the list of available MCP tools provided by the Velocity MCP Server.

> **Note:** The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session expires and you are required to re-authenticate.

Results

You have successfully connected to the Velocity MCP server by using Claude Desktop.

What to do next

You can run prompts related to Velocity to confirm a successful connection.

Connecting to the Measure MCP server by using GitHub Copilot in Visual Studio Code

You can connect to the Measure Model Context Protocol (MCP) server by using GitHub Copilot in Visual Studio (VS) Code to interact with Measure and retrieve the necessary data and insights.

Before you begin

You must have completed the following tasks:

- Installed Measure and configured Key cloak. See Prerequisites to connect with the MCP server in Measure on page 1827.
- Configured OIDC authentication in Measure. See Authenticating users with OpenID Connect (OIDC) on page 1817
- Obtained the MCP Server URL. For example: `https://<velocity_hostname>/mcp`
- Ensured that you have installed VS Code and the GitHub Copilot extension.

1. **Optional:** Open the terminal and enter the following command to allow the self-signed certificates if you are using a self-signed certificate during the Measure installation.

   ```
   export NODE_TLS_REJECT_UNAUTHORIZED=0
   ```
2. Enter the following command to open VS Code.

   ```
   code .
   ```

3. Open the **Command Palette** by pressing the keys Ctrl+Shift+P or Cmd+Shift+P.
4. Search and select **MCP: Add Server**.
5. Choose **HTTP** as the server type, when prompted for the transport type.
   The Velocity MCP Server uses HTTP-based transport.
6. Enter the **MCP Server URL** in the **URL** field.
   VS Code automatically generates the required `mcp.json` configuration.
7. Enter a name for the server. For example: Velocity MCP Server.
8. Select **Workspace**, when VS Code prompts you to choose where to save this configuration.
   - **Global** – used for all VS Code workspaces
   - **Workspace** – used only in the current folder/project

     Note: When **Workspace** is selected, a `.vscode` folder is automatically created, and an `mcp.json` file is generated within it.

     A sample `mcp.json` file is as shown below:

```
{
 "servers": {
  "velocity-mcp-server": {
   "url": "<velocity_mcp_server_url>",
   "type": "http"
  }
 },
 "inputs": []
}
```

Property	Description
<velocity_mcp_server_url>	Specifies the full Velocity MCP server URL. For example: `https://<velocity_hostname>/mcp`

   Result

   The `mcp.json` file is opened.
9. Click **Start** for the added MCP server.

   Result

   A prompt appears to authenticate via OAuth.
10. Click **Allow**.

    Result

    The following prompt is displayed:

```
Do you want code to open the external website?
```

11. Click **Open**.

Result

The OIDC login page is displayed.

12. Sign in using the Measure OIDC user credentials.

    Result

    A consent screen is displayed requesting access to MCP tools, Measure APIs (used internally for tool execution), and basic profile and role information.

13. Click **Yes**.

    You are automatically redirected to VS Code or you can also close the browser and switch to VS Code.

14. Verify the connection.

    The Measure MCP Server must display **Running** as the status.

> **Note:** The MCP server times out after 30 minutes of inactivity. If no tool calls occur during this period, your session expires and you are required to re-authenticate.

Results

You have successfully connected to the Measure MCP server by using GitHub Copilot in VS Code.

What to do next

You can run prompts related to Measure to confirm a successful connection.

Value Stream Management

A value stream represents the objects and activities that you employ to deliver customer value, from initial concept to final delivered project.

The value stream provides a strategic window into your project workflow and enables you to drill-down to individual objects. The objects, such as builds, pull requests, issues, and tests, come from tools that you integrate into the value stream. Individual objects are represented graphically by small circles called dots. Dots frequently provide information from logically related tools, such as issues linked to source control commits. You filter the value stream by building queries on page 3117 with the *DevOps query language*.

About value streams

Value streams are displayed on the **Value stream** page in individual cards that provide status information and links to the value streams. You can change the page to a table view and customize the rows with metric data taken from the value streams.

When you create a value stream, a minimal value stream is configured and displayed on the **Value stream** view. The highest level of a value stream is a *phase*. Phases represent important, organizational parts of the value stream, such as "Planning," or "Development." On the **Value stream** view, processing is done in left-right order. For example, your first phase might be for planning and contain items created in an issue tracking system. Your next phase, used for development, can track source control activity related to the issues, and any builds triggered by the activity. Your final phase, used for deployment, might track the related build artifacts as they move through your testing environments toward production.

Phases contain *stages* that define process flow within a phase. A development phase that integrates a source control tool might contain an *In Progress* stage followed by *In Review* and *Merged* stages. When you customize a value stream, you define the phases and stages and their order.

Stages are containers for *dots*. Dots represent units of work from Measure or tools that are integrated into the value stream. Git commits or Jira issues, to take just two, are represented by individual dots. Work items, such as commits and builds, can be combined into individual dots. A dot's position in the value stream conveys important information about the object. Dots in a stage named *Merged* might represent items merged into Git repositories. An item's shape and color convey information about the item's type and status. A dot outlined in red, for example, might mean the item is past schedule. When dots change state, they move to new stages in near-real time. Finally, when you click on a dot, the displayed card provides information about the work items, including their history, and provides links to associated tools.

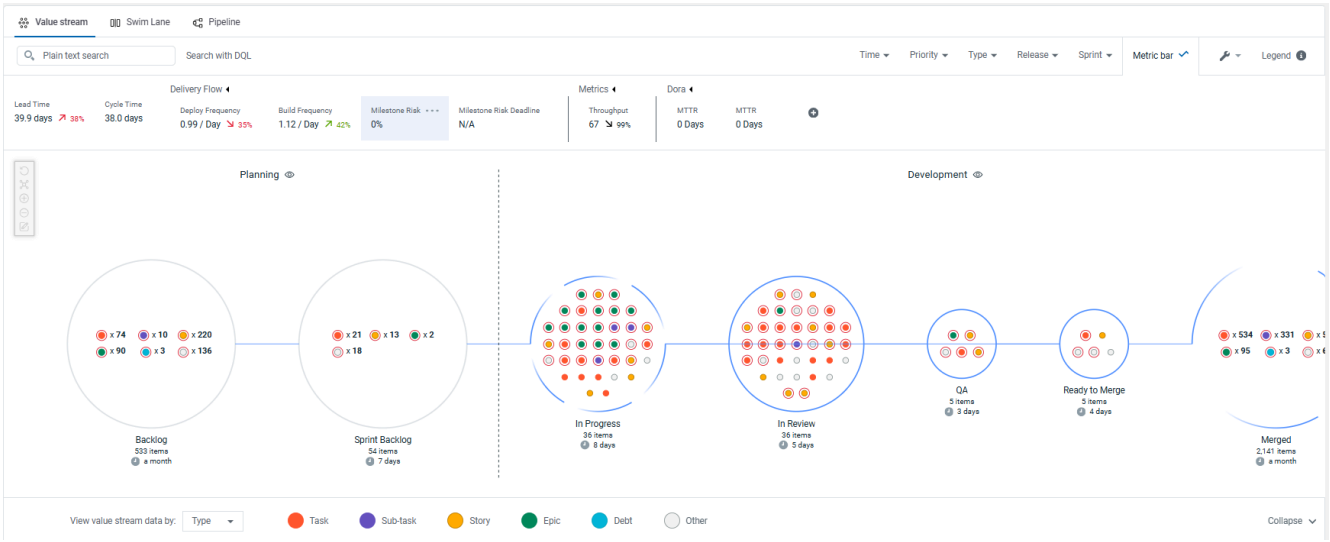
Customizing the value stream

When you create a value stream, default stages and phases are configured, but until you integrate the tools that you use with Measure , the stages will not contain items. You add your tools to a value stream by defining plugin integrations for them. After you integrate a tool, its items, such as pull requests, can be displayed in the appropriate stages. Measure provides plugins for an ever-growing number of 3rd-party tools and services that you can integrate into your value stream.

The JSON file associated with the value stream defines the phases and stages. The value stream default phases are Planning, Development, and Deployment. You can modify the JSON file to display the phases and stages that you normally use in your work.

The following figure shows a typical user-configured value stream. Notice that items are filtered by *Type*. For information about how to manage data displayed in the **Value stream** view, see Navigating value streams on page 3103.

Figure 7. User-configured value stream



Integrating tools into the value stream

You use the same JSON file to create integrations that you use to customize the value stream. After you define an integration, items associated with the integrated tool can appear in the stages on the **Value stream** view, and the integration is listed on the **Integrations** page in the web UI.

You can associate integrated tools by defining link rules for them on page 3066. If you link an issue tracking tool with a source control management tool, for example, you can access information about the linked tools from affected dots.

For information about the value stream integrations, see About value stream integrations on page 3078.

Querying the value stream

To see objects from your integrated tools, add queries to the value stream. You use the Value Stream Query language to filter the information displayed on the value stream. You build simple queries by combining searchable items with operators and values. A searchable item is a query-able object, such as `issue.status`. Queries can be scoped to the entire value stream or a stage. Each stage can have a query. You can create queries on the **Value stream** view, or define them in the value stream JSON file.

For information about the DevOps query language, see DevOps query language (DQL) on page 3117.

Swim lanes

Swim lanes provide a user-based perspective of your value stream data. When you use the swim lanes feature, value stream data is organized into a table where the columns represent value stream stages, and the rows represent users.

For information about swim lanes, see Managing swim lanes on page 3113.

About pipelines

When you create a value stream, a corresponding pipeline is configured. A pipeline is a sequence of stages that retrieve input and run applications and jobs. Pipelines accept input from source control management (SCM) repositories or other external applications, such as HCL DevOps Deploy (Deploy). You add applications to the pipeline from external tools that you integrate with Measure, such as Jenkins, and IBM® DevOps Build. You can run applications, releases, and deployments with the **Pipeline** view.

For information about pipelines, see Pipelines on page 3127.

Creating value streams

Create a value stream and then customize it by defining phases and stages that reflect your typical workflow.

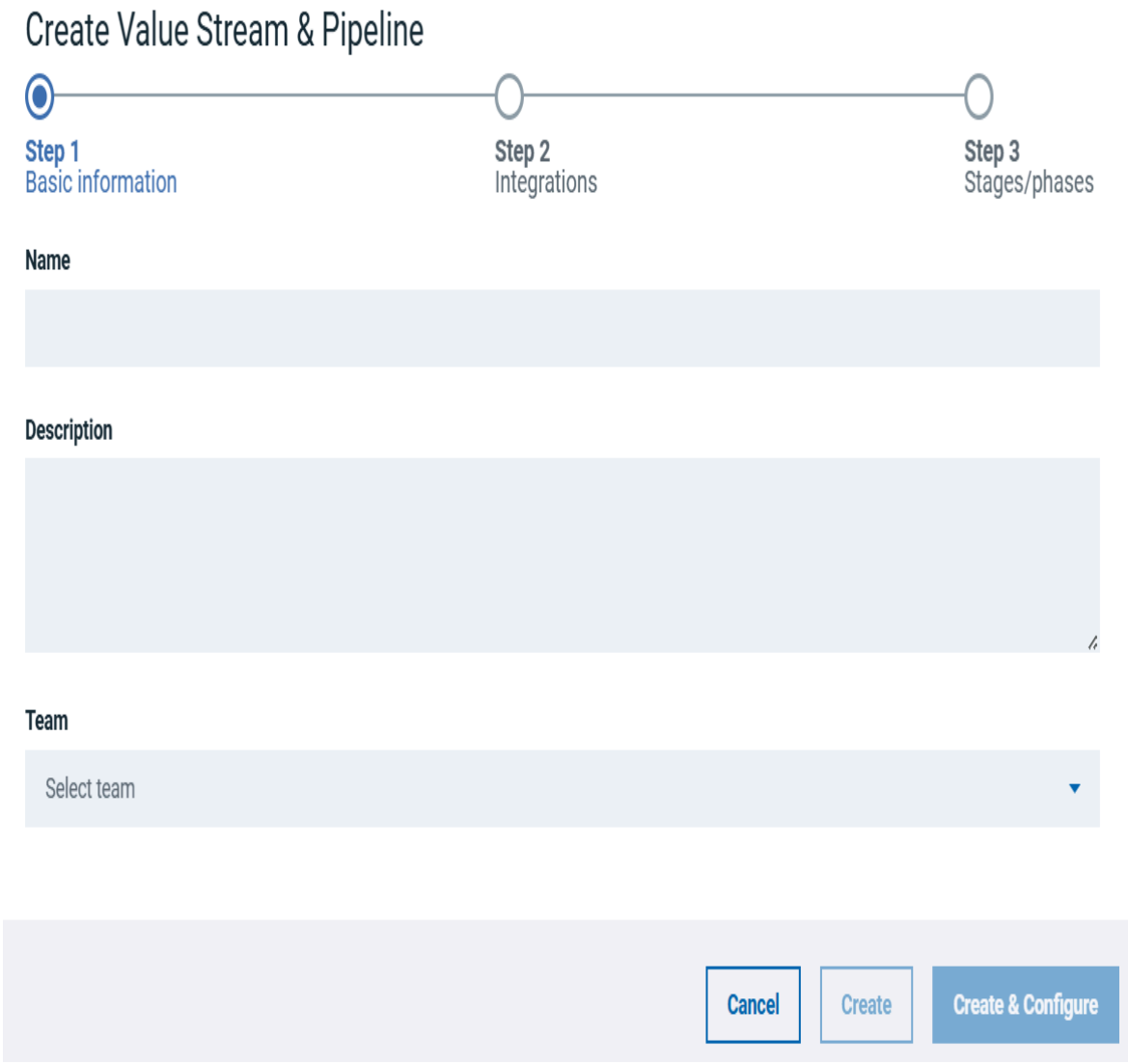
About this task

When you create a value stream, a plain value stream is displayed on the **Value Streams** view. To modify the value stream and integrate external tools, edit the `value_stream_name-vsm.json` file attached to the value stream.

1. To create a value stream and configure, perform the following steps:

    a. On the Measure dashboard, click **Value Streams**, and then click the **Create value stream**.

    b. The **Create Value Stream & Pipeline** window is displayed with three steps: **Basic information** about the value stream, **Integrations** to use in value stream, and **Stages/phases** for value stream.



    c. On the **Create Value Stream** window, under the **Step 1**, in the **Name** field, enter a name for the value stream.

    d. In the **Description** field, enter a description for the value stream.

    e. In the **Team** list, select a team to manage the value stream.
    The teams that are available depend on the permissions granted to your user role. In general, you can select teams to which you belong, or that you administer.

    f. Perform any of the following actions:

- To create a value stream and configure the integration later, click **Create** and go to 2 on page 3054.
- To create and configure a value stream, click **Create & Configure** and go to 1.g on page 3053.

g. Under the **Step 2**, in the **Application Lifecycle Management Integration** field, select the required application lifecycle management integration for the value stream.

> **Note:** You can select **Click here to enable a new integration** to configure a new application lifecycle management integration.

h. In the **Source Code Management Integration** field, select the required source code management integration for the value stream.

> **Note:** You can use **Click here to enable a new integration** to configure a new source code management integration.

i. Click **Next**.

j. In the **Step 3**, under the **PLANNING** section, in the **Select the Backlog stage statuses:** field, select the required status of the application lifecycle management integration to displayed in backlog stage for the value stream.

k. In the **Select the To Do stage statuses** section, select the required status of the application lifecycle management integration to displayed in To do stage for the value stream.

l. Under the **DEVELPOMENT** section, in the **Select the In Progress stage statuses:** field, select the required issue status of the application lifecycle management and select the required pull request statuses of the source code management integration for the value stream.

m. In the **Select the In Review stage statuses:** field, select the required issue status of the application lifecycle management and select the required pull request statuses of the source code management integration for the value stream.

n. Under the **COMPLETED** section, in the **Select the Completed stage statuses:** field, select the required issue status of the application lifecycle management and select the required pull request statuses of the source code management integration for the value stream.

o. Click **Save**.
The new value stream is displayed on the **Value Streams** page in the **All Value Streams** tab.

p. **Optional:** To add a created value stream to **My Favorites**, click the **All Value Streams** tab and then click to display the value streams list view. In the list, locate the **Name** of value stream to add to **My Favorites** and then click on the left side of the field.

The value stream is listed on the **My Favorites** tab.

q. **Optional:** To remove a value stream from **My Favorites**, click ★ and the value stream is removed from the **My Favorites** tab.

2. Click and select **Edit value stream map** to modify the JSON file in the code on page 3067 or tree on page 3071 view editors.

   You can also click and select **Download value stream map**.

   Result

   The `value_stream_name-vsm.json` file is downloaded to the specified location.
3. Using a text editor, modify the `value_stream_name-vsm.json` by completing the following steps:

   a. **Optional:** Customize phases and stages. on page 3062

   You can modify the number of phases and stages, and add queries and gates to stages. A query defined in the heading sets the scope for the entire value stream. Queries defined for a stage are scoped to the stage.

   b. Define integration objects that integrate external tools into the value stream. on page 3077

   c. Define lead- and cycle-times for the value stream. on page 3064

   Lead-time refers to the time a project, represented by a value stream, takes from initial conception to final disposition. Cycle-time refers to the amount a time individual objects take to complete. Lead- and cycle-times can be displayed on the **Value stream** tab.

   d. Define link rules for integrated tools. on page 3066

   Link rules define how the tools integrated into the value stream are related.

   e. Define metrics for the value stream on page 3084.

   f. Define metrics for the metrics bar on page 3084.

   g. Define gates for the value stream. on page 3059

   h. Define mapping rules for the value stream. on page 3064

4. Click , and select **Edit value stream map** > **Import JSON**, and then select the modified file.
5. To modify the integrations in the existing value stream, perform the following steps:

   a. Open the value stream for which you want to modify the integrations.

   b. Click and select **Manage integrations**.

   Result

   The **Edit Value Stream** dialog is displayed.

c. In the **Application Lifecycle Management Integration** field, remove the existing integration and select the required application lifecycle management integration from the list.

Note: You can use **Click here to enable a new integration** to configure a new application lifecycle management integration.

d. In the **Source Code Management Integration** field, remove the existing integration and select the required source code management integration from the list.

Note: You can use **Click here to enable a new integration** to configure a new source code management integration.

e. Click **Save**.

Result

The modified integration details are updated in the `value_stream_name-vsm.json` file and the value stream displays the data from the modified integrations.

What to do next

Next, filter value streams by creating queries with the DevOps query language on page 3117. Learn about the information presented by value streams. on page 3103

Setting up a stage

Setting up a stage with queries you define in the search bar work within the scope set for the value stream.

Before you begin

Set up a value stream stage with defined queries that you need for your project.

About this task

Value stream queries determine on page 3117 the items that are displayed in the value stream stages. Queries that you define in the search bar work within the scope set for the value stream. The scope is set in the `value-stream-name.json` file. The scoping levels are detailed below.

1. Only items from integrations defined for the value stream are available. Define the integrations in the `value-stream-name.json` file.
2. Queries defined at the value stream level. To set this scope, define a query in the heading of the `value-stream-name.json` file.
3. Stage level. To set this scope, define a query in a stage object of the `value-stream-name.json` file.

Results

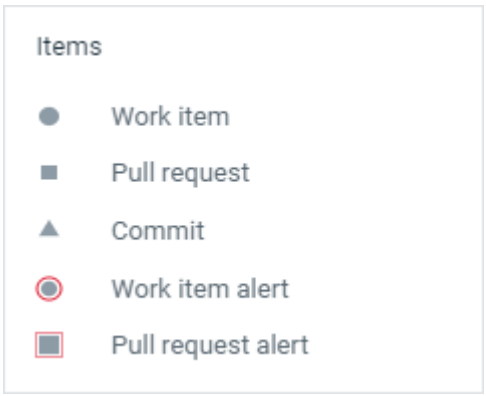
Defined queries are set up for the value stream.

Example

Queries that you define on the **Value streams** view search bar, work within the scope set for the value stream. For example, in the `value-stream-name.json` file you might set the scope by integrating a shared Jira® instance and defining a stream-level query that restricts issues to just those that belong to your project. In the In Progress stage, you might define a query that matches issues with a status of `In progress`. When the user opens the value stream, only items that match the pre-defined scope appear in the In Progress stage. You can then define a query in the search bar to additionally filter the items, perhaps by matching a particular issue ID or name.

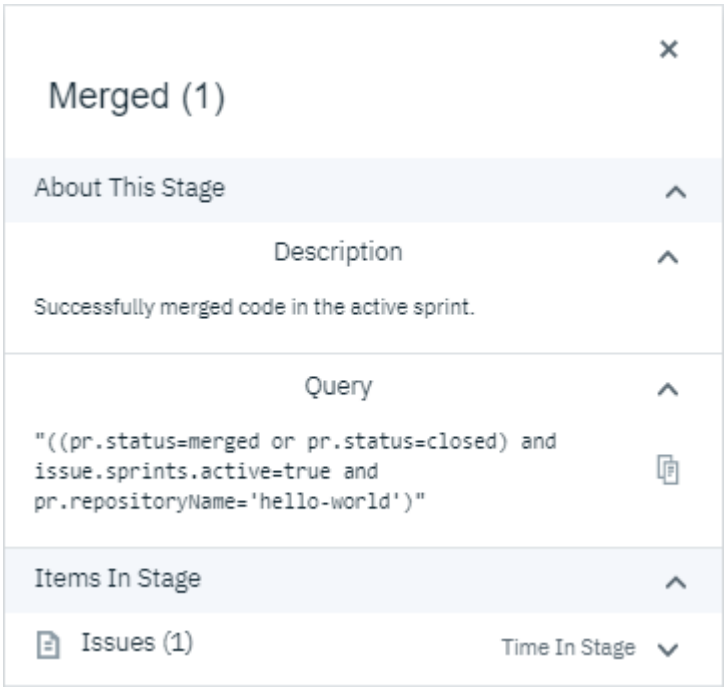
When you open a value stream, items that match the default queries are displayed in their appropriate stages. An items shape indicates what type of item is represented, such as commit or work item. The **Legend** drop-down list describes what the shapes represent, as shown in the following graphic:

Figure 8. Item types



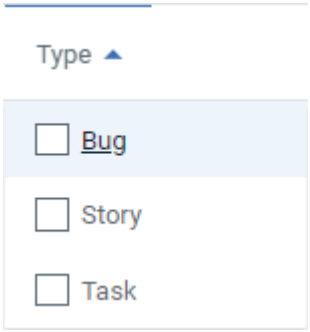
To see the default query defined for a stage, click the mouse on it and then click **About this Stage** on the pop-up that is displayed. The following graphic shows a pop-up for a stage called Merged that contains one item (not shown).

Figure 9. Stage summary



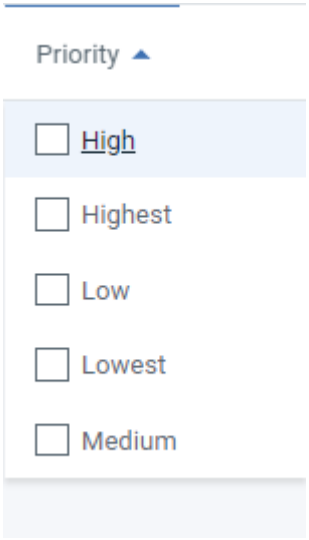
In addition to queries, you can apply other filters to the value stream. By default, when you open a value stream, items are filtered by type. When you click the **Type** drop-down, it lists the issue types, as shown in the following graphic:

Figure 10. Issue types



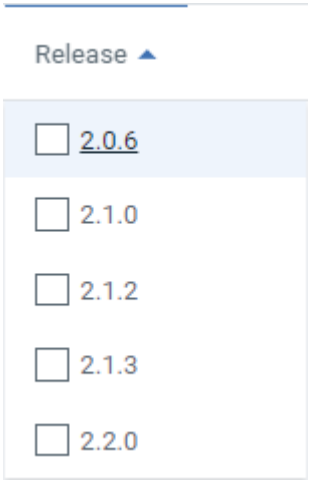
You can also filter items by priority. When you click the **Priority** drop-down, it lists the priority types, as shown in the following graphic:

Figure 11. Priority types



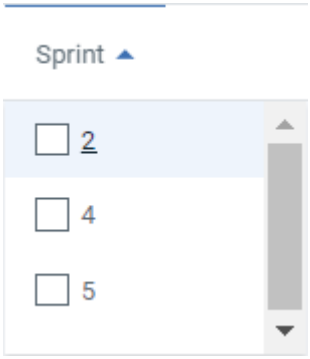
You can also filter items by release. When you click the **Release** drop-down, it lists the release types, as shown in the following graphic:

Figure 12. Release types



You can also filter items by sprint. When you click the **Sprint** drop-down, it lists the sprint types, as shown in the following graphic:

Figure 13. Sprint types



What to do next

Next, filter value streams by creating queries with the DevOps query language on page 3117. Learn about the information presented by value streams. on page 3103

Customizing value streams

Customize the `json` file to reflect your typical workflow.

When you create a value stream, default stages and phases are configured and displayed in the **Value Stream** tab. The structure of the value stream is determined by the `json` file attached to the value stream. You can customize the `json` file by using the following methods:

- VSM JSON code editor on page 3067
- VSM JSON tree editor on page 3071
- Download the file, modify it, and then reattach it to the value stream by uploading it to Measure.

A value stream `json` file consists of the following areas:

- The **heading** identifies the value stream.
- The **phases** defines the value stream's phases and stages. It contains a query that filters the dots appearing in the stage.
- The **leadTimeChange** is the quantity of time it takes a dot to progress from commit to production deployment. The metrics calculation is based on the date of the commit and date of the deployment into production. The lead time for changes metrics is a default metric added to all available value streams metrics bar.
- The **leadTime** defines the lead-time metric. In general, lead-time reflects the time from the start of a project until its final disposition.
- The **cycleTime** defines the cycle-time metric. In general, cycle-time reflects the time project elements take to resolve or complete.
- The **devcycleTime** defines the development cycle-time metric. In general, dev-cycle time reflects the time project elements take to reach from development in-progress to development complete (before the PROD stage).
- The **mapping** defines status levels for issue tracking systems.
- The **integration** defines integration with external tools. The default stages are empty until you integrate tools into the value stream. You add your tools to a value stream by defining integration objects on page 3077.

- The **Metrics** provides metric data to the value stream.
- The **metricsBar** section displays metrics data in the value stream **Metric bar**.
- The **linkRules** defines relationships between integrated tools.

The following code block shows the default value stream `json` file:

```
{
  "_id": "velocity-generated-value",
  "pipelineId": "velocity-generated-value",
  "tenantId": "velocity-generated-value",
  "description": null,
  "query": "",
  "phases": [
    {
      "name": "Planning",
      "description": "",
      "stages": [
        {
          "name": "Backlog",
          "query": "",
          "description": "",
          "wipLimit": null,
          "showAlerts": false,
          "gates": null
        },
        {
          "name": "Queue",
          "query": "",
          "description": "",
          "wipLimit": null,
          "gates": null
        }
      ]
    },
    {
      "name": "Development",
      "description": "",
      "stages": [
        {
          "name": "In Progress",
          "query": "",
          "description": "",
          "wipLimit": 10,
          "gates": null
        },
        {
          "name": "In Review",
          "query": "",
          "description": "",
          "wipLimit": 5,
          "gates": null
        },
        {
          "name": "Merged",
          "query": "",
          "description": "",
          "wipLimit": null,
```

```
          "gates": null
        }
      ]
    },
    {
      "name": "Deployment",
      "description": "",
      "stages": [
        {
          "name": "DEV",
          "query": "",
          "description": "",
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "QA",
          "query": "",
          "description": "",
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "PROD",
          "query": "",
          "description": "",
          "wipLimit": null,
          "gates": null
        }
      ]
    }
  ],
  "leadTime": {
    "start": "Backlog",
    "end": "PROD"
  },
  "cycleTime": {
    "start": "In Progress",
    "end": "PROD"
  },
  "devCycleTime": {
    "start": "In Progress",
    "end": "In Review"
  },
  "mappings": {
    "priority": {
      "Lowest": [
        "Lowest"
      ],
      "Low": [
        "Low"
      ],
      "Medium": [
        "Medium"
      ],
      "High": [
        "High"
      ],
```

```
      "Highest": [
        "Highest"
      ]
    }
  },
  "integrations": [],
  "linkRules": [],
  "metrics": [
    {
      "name": "Work Item Type",
      "repeatEvery": "1 minute",
      "repeatAt": null,
      "metricDefinitionId": "WORK_ITEM_TYPE",
      "properties": {
        "query": null,
        "field": "issue.type"
      }
    }
  ],
  "metricsBar": null
}
```

Heading

The heading properties are common to all value streams.

Table 1149. Properties for stages object

Property	Description
_id	Value stream identifier. The value is generated by Velocity Measure.
pipelineId	Identifier of the pipeline associated with the value stream. The value is generated by Measure.
tenantId	Tenant identifier. The value is generated by Measure.
description	Value stream description.
query	A query that sets the scope to the entire value stream, not just a stage. If you share a Jira instance, for example, you might define a query that restricts the value stream to a single team's issues.

Phases

The `phases` JSON array defines the value stream phases and stages. The single `phases` array for each value stream contains `stages` objects that define the stages for the containing phase. A `phases` array can contain any number of `stages` objects.

The following code fragment shows a Planning phase that contains stages named Backlog Scrum, and Selected for Development.

```
"phases": [
    {
      "name": "Planning",
      "description": "Planning stage",
      "stages": [
        {
          "name": "Backlog Scrum",
          "query": "issue.status=\"Backlog\"",
          "description": null,
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "Selected For Development",
          "query": "issue.status=\"TO DO\" or issue.status=\"To Do\"",
          "description": null,
          "wipLimit": null,
          "gates": null
        }
      ]
    },
```

Table 1150. Properties for stages object

Property	Description
name	Stage name
query	The DevOps query that defines which dots are displayed in the stage. In general, query parameters reference the tools integrated into the value stream. See the examples later in this topic.
description	Stage description
wipLimit	Represents the 'work in progress' limit. If the number of stage items exceeds the limit, a warning is displayed. To disable warnings, leave the value blank.
showAlerts	Alerts for change in status of dots. By default, the `showAlerts` property is set to `false` for Backlog and Production stages and `true` for any other stages. You can set the `showAlerts` property for any stages. Possible values are `true` and `false`.
showSpeed	Defines whether to show the stage speed or not. By default, the `showSpeed` property is set to true, allowing you to view the stage speed through the visual representation of the stage circle. Click the **Legend** on the **Value streams** page to view the stage speed corresponding to the stage circle representation. You can set this property to false if you don't

Table 1150. Properties for stages object (continued)

Property	Description
	wish to see the stage speed. For more information about the stage circle speed, seeNavigating value streams on page 3103.
gates	Gates determine if objects are valid for a given context. Valid objects are referred to as passing the gate; invalid ones are referred to as failing the gate. Valid objects can appear in value streams.

Lead time, cycle time, and dev-cycle time metrics

The lead time, cycle time, and dev-cycle time properties provide useful metrics about value stream performance. The lead-time property reflects the elapsed-time from project start through final disposition. The cycle time property reflects the time that individual project elements take to complete.

Configure the metrics by assigning a starting stage to the `start` property, and an ending stage to the `end` property. For example, if you set `start` to the `Backlog` stage, items are tracked when they move from the `Backlog` stage to the next stage. To carry the example forward, if you set `end` to the `Deployed` stage, items are finished when they move into the `Deployed` stage.

When you configure the metrics, the calculated values are displayed on the **Value Stream** tab in the **Metric bar**.

The following code example illustrates typical lead- and cycle-time configurations.

```
    "leadTime": {
      "start": "Backlog",
      "end": "Deployed",
    },
     "cycleTime": {
      "start": "In Progress",
      "end": "Merged",
    },
    "devCycleTime": {
      "start": "In Progress",
      "end": "In Review"
  },
```

Mapping

The mappings object defines status values for issue tracking systems. The following code example illustrates the default mappings:

```
"mappings": {
    "priority": {
      "Lowest": [
        "Lowest"
      ],
      "Low": [
        "Low"
```

```
      ],
      "Medium": [
        "Medium"
      ],
      "High": [
        "High"
      ],
      "Highest": [
        "Highest"
      ]
    }
  },
```

Query examples

A DevOps query determines the dots that are displayed. A query at the value stream level filters, or scopes, the entire value stream. A query at the stage level filters the stage, and a query at the gate level filters the gate. Gates are discussed later in this topic.

In general, query parameters reference the tools integrated into the value stream or applications that belong to the associated pipeline. For a Jira integration you might restrict dots appearing in a stage named "In progress" with the following query:

```
"query":"issue.status="In Progress"
```

For a Git integration you can restrict dots appearing in a stage named "Merged" with the following query:

```
"query":"(pr.status=merged OR pr.status=closed) AND deployment.env!=DEV AND deployment.env!=QA AND deployment.env!=PROD and build.status!=success"
```

The following code fragment shows a planning phase with several stages and the corresponding queries.

```
{
      "name": "Development",
      "description": null,
      "stages": [
        {
          "name": "In Progress",
          "query": "issue.status=\"In Progress\" and build.status!=success",
          "description": null,
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "Review",
          "query": "issue.status!=Done and pr.status=open",
          "description": null,
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "Merged",
          "query": "(pr.status=merged OR pr.status=closed) AND deployment.env!=DEV AND deployment.env!=QA AND
 deployment.env!=PROD and build.status!=success",
```

```
          "description": null,
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "Build",
          "query": "build.status=success AND deployment.env!=DEV AND deployment.env!=QA AND
deployment.env!=PROD",
          "description": null,
          "wipLimit": null,
          "gates": null
        }
      ]
    },
```

For more information, see the DevOps query language reference. on page 3117

Link rules

Link rules define how integrated tools are related to one another on page 3080. When tools are linked, objects that appear in a value stream provide information and links to the associated tools. You might link an issue tracking system, for example, to a source control tool

When links are established, you can access information about the linked tools from the affected dots in the value stream. In general, you can define linking rules for any integrated tools, and define multiple linking rules in a single value stream.

Metrics

You can provide metric data to your value stream by querying the data integrated into Measure. The data can originate from plugin integrations, or REST API calls from external clients. DevOps Query Language (DQL) queries can be defined in the `value_stream_name-vsm.json` file.

You can use the default `Work Item Type` metric definition as a template for your metrics.

Metric bar

You can provide metrics data to value streams and display the data in charts and graphs attached to the stream on page 3084. To enable the **Metric bar**, add `metricsBar` definitions to the `value_stream_name-vsm.json` file attached to the value stream.

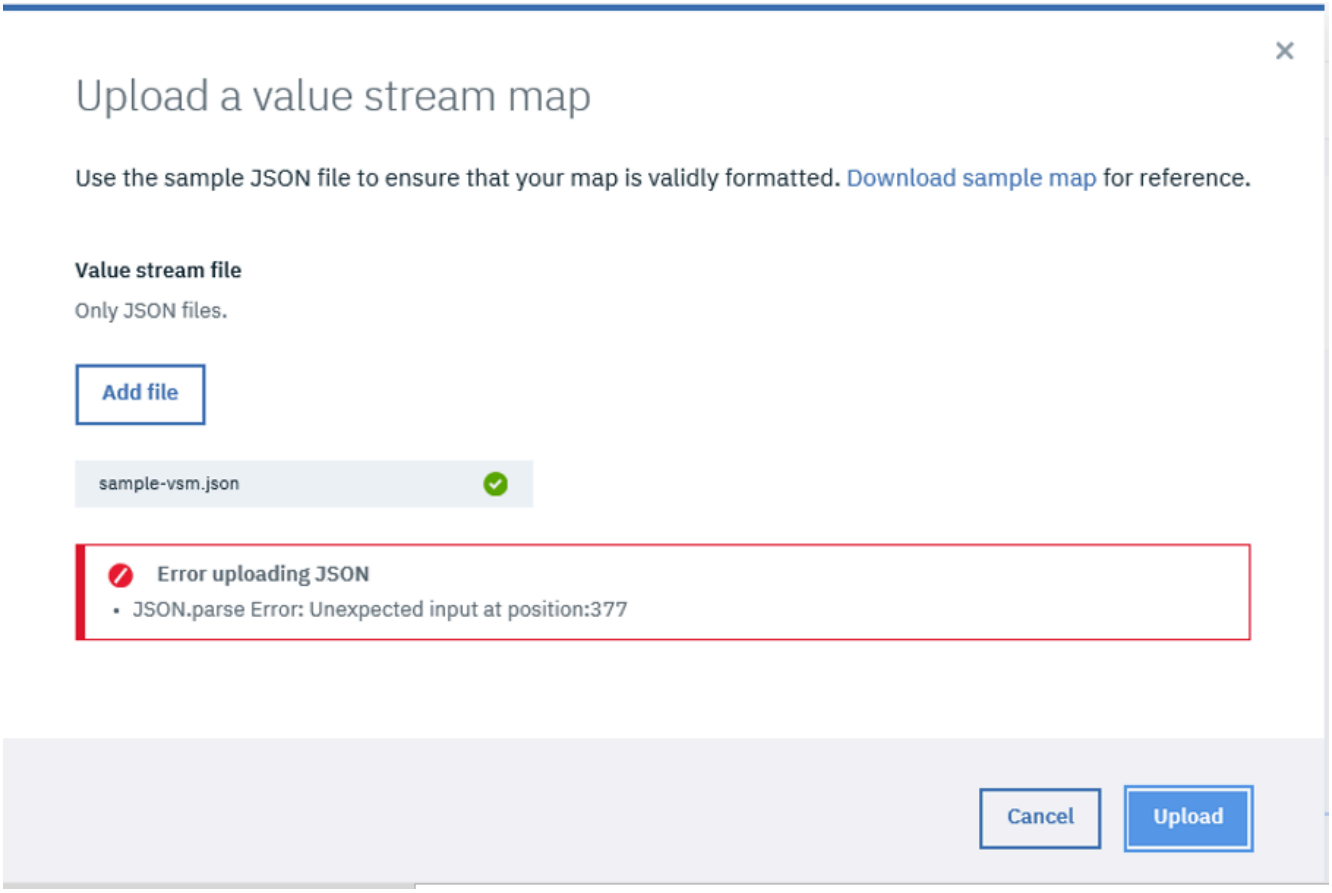
Integrations array

You provide data from external tools to value streams by defining integration objects on page 3077. For information about the supported plugins, see #unique_3272.

Troubleshooting

The figure below displays the error message when the `value_stream_name-vsm.json` file with invalid syntax is uploaded. The error message also displays the `.json` parse error position in the `value_stream_name-vsm.json` file. The system will generate similar error messages whenever an invalid `value_stream_name-vsm.json` file is uploaded.

Figure 14. Error message



VSM JSON code editor

The code editor allows you to directly edit the value stream JSON file from the UI.

About this task

Use the online code editor to edit value stream JSON files. Starting with Measure version 3.0, you no longer need to download the JSON and edit a local copy. Now you can edit value stream JSON files within the product web interface. On the **Value Streams** page, use the VSM JSON code editor to edit and save the value stream workflow file. In code view, the editor allows you to perform the following:

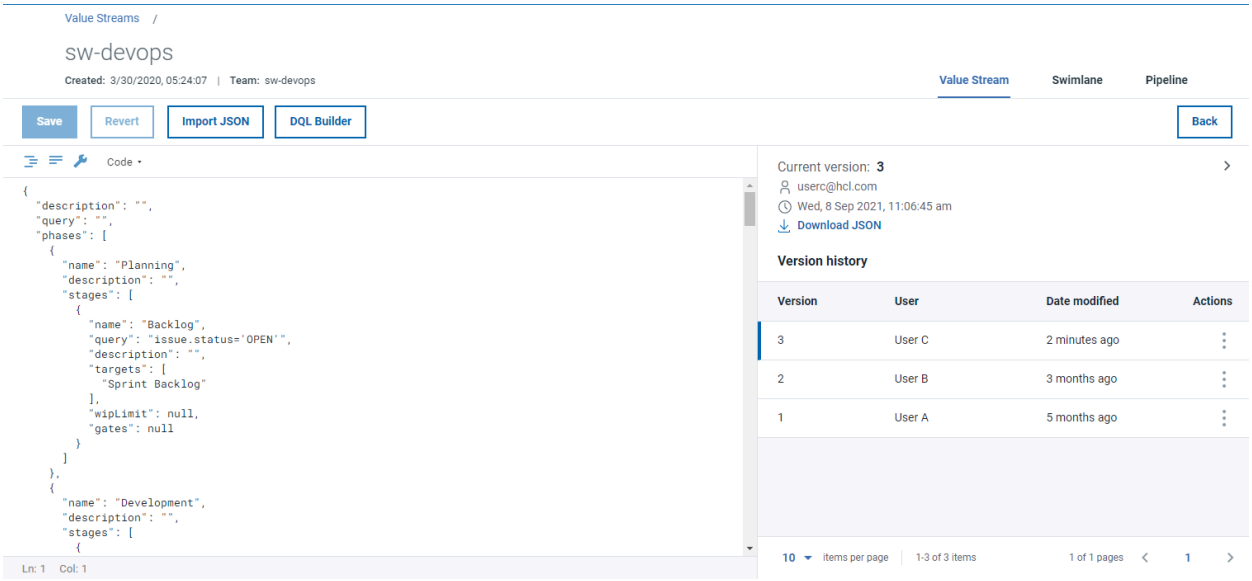
- Create new versions
- Edit versions
- Review versions
- Import and download versions
- View JSON file contents including creator, version number, and date created

Additionally, you can format, compact, repair JSON data, and save with overwrite protection for simultaneous user editing scenarios.

1. On the **Value Streams** page, click the required value stream.
2. Click and select **Edit value stream map** to open the VSM JSON code editor.

   The JSON code editor is shown in the figure below.

   Figure 15. JSON code view



Note: The JSON code editor is the default view mode when you open the editor.

Note: In the **Version history** table on the right side of the display, the current version of the JSON shown on the left side of the display is indicated by a vertical blue bar next to the **Version** field in the table and is the file displayed and open in the editor.

Note: The icons on the left side of the display above the editor include the following:

- Click to format the JSON data with proper identification and line feeds.
- Click to compact the JSON data by removing all whitespaces in the JSON file.
- Click to repair the JSON file including fixing quotation marks and escape characters, removing comments and JSON with padding (JSONP), and converting JavaScript objects into JSON.

- On the right of the icon above the code editor, you can click and select the **Code** or **Tree** on page 3071 view drop-down to choose between a plain text editor or a user-friendly editor respectively.

  Note: When you select **Edit value stream map** to open the editor, the editor is always opened in **Code** view by default.

3. To import, download, view, and edit a JSON file in the editor, perform either of the following:
   - Click **Import JSON**, select the JSON file, and click the **Open** button.

     Note: The file is opened and displayed in the editor.

   - On the right side of the display above the **Version history** table, click **Download JSON** to download the latest JSON file revision shown in the **Current version** field above the creator.

     Note: Above the **Version history** table, you can view detailed information for the latest JSON file with fields including **Current version**, creator, and file creation day, date, year, and time.

   - In the **Version history** table with fields including **Version**, **User**, **Date modified**, and **Actions**, locate the required JSON file version and in the **Actions** field, click and select either **View code** or **Download** according to your requirement.

     If you want to use a previous version JSON file as the current version, perform the following:

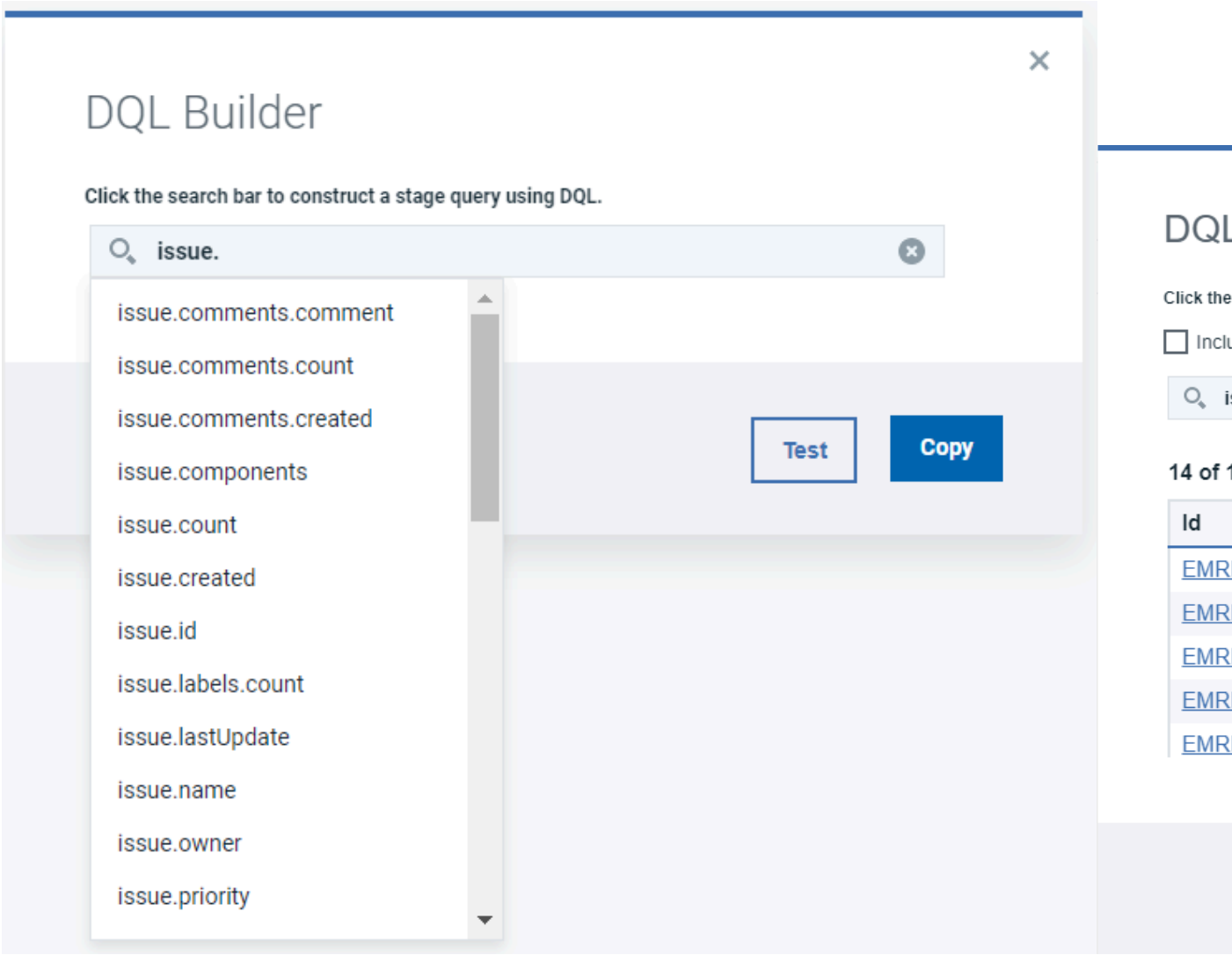
     - In the **Version history** table, click for the required previous version and select **Download**.
     - Click **Import JSON** and navigate to the downloaded file.
     - Click **Open**.
     - Click **Save**.

       Note: After a JSON file is successfully saved, you cannot revert the changes that were made prior to saving the file. If you want to use a previous version JSON file as the current version, use the aforementioned procedure.

     At the bottom of the **Version history** table, you can perform and view the following:

     - In the **items per page** drop-down, you can select **10**, **20**, **50**, or **100** as the number of items to display per page in the **Version history** table.
     - View number of items listed in the table versus total number of items. For example: 1-3 of 3 items.
     - View page number versus total number of pages.
     - View the page number currently displayed in the **Version history** table and click **<** or **>** to go to the previous or next page respectively.

4. To create a query, click **DQL Builder** and then **DQL Builder** window appears. Type the query in search field and select the required one from the list to create query. After creating query, click **Test** to view results for the selected query. Click **copy** to copy the created query. You can use the copied query in JSON code or tree editor.



5. On the left side of the display in the file editor, perform the required editing in the file and click **Save** to capture your changes.

Note: If you make any changes before saving the file, you can click **Revert** to undo all your changes.

Result

When you click **Save**, a notification is displayed containing one of the following messages:

- If all the changes you made to the JSON file are validated when saved, a **Success** notification with a vertical green bar is displayed upon completion.
- If an error is associated with the file, a notification with a vertical red bar is displayed with either of the following information:

- The source of error.
- Selected file type is not supported.

6. Overwrite protection provides visibility to users that attempt to simultaneously edit the same workflow. The following example details the procedure for two users simultaneously attempting changes to the same file. User 1 creates and saves version 78 by editing version 77 and User 2 edits version 77 and attempts to save the file with the result of displaying the **OVERWRITE PROTECTION Confirm changes? Users have made 1 new version since loading this page. Do you want to overwrite the latest changes?** modal prompting to click either the **View latest changes** or **Confirm** buttons detailed in the following table:

Button	Description
View latest changes	Modal closes and refreshes the **Version history** table with new saved version 78 and the text editor displays the changes made by User 2 for this example.
Confirm	Saves the changes to version 77 and selects the same version in the **Version history** table for this example.

What to do next

Documentation for the VSM JSON tree editor is found here on page 3071.

VSM JSON tree editor

The tree editor allows you to directly edit the value stream JSON file from the UI.

About this task

Use the online tree editor to edit value stream JSON files. Starting with Measure version 3.0, you no longer need to download the JSON and edit a local copy. Now you can edit value stream JSON files within the product web interface. On the **Value Streams** page, use the VSM JSON tree editor to edit and save the value stream workflow file. In tree view, the editor allows you to perform the following:

- Change **Type**, **Insert**, or **Append** the **Auto**, **Array**, **Object**, or **String** field types defined in the following:
  - **Auto** is a field type based on the value of a string, number, boolean, or null.
  - **Array** contains an ordered collection of values.
  - **Object** contains an unordered set of key and value pairs.
  - **String** is a field type not based on the value and returned as a string.
- **Duplicate** fields
- **Extract** objects and arrays
- **Remove** fields

Additionally, you can format, compact, repair JSON data, and save with overwrite protection for simultaneous user editing scenarios.

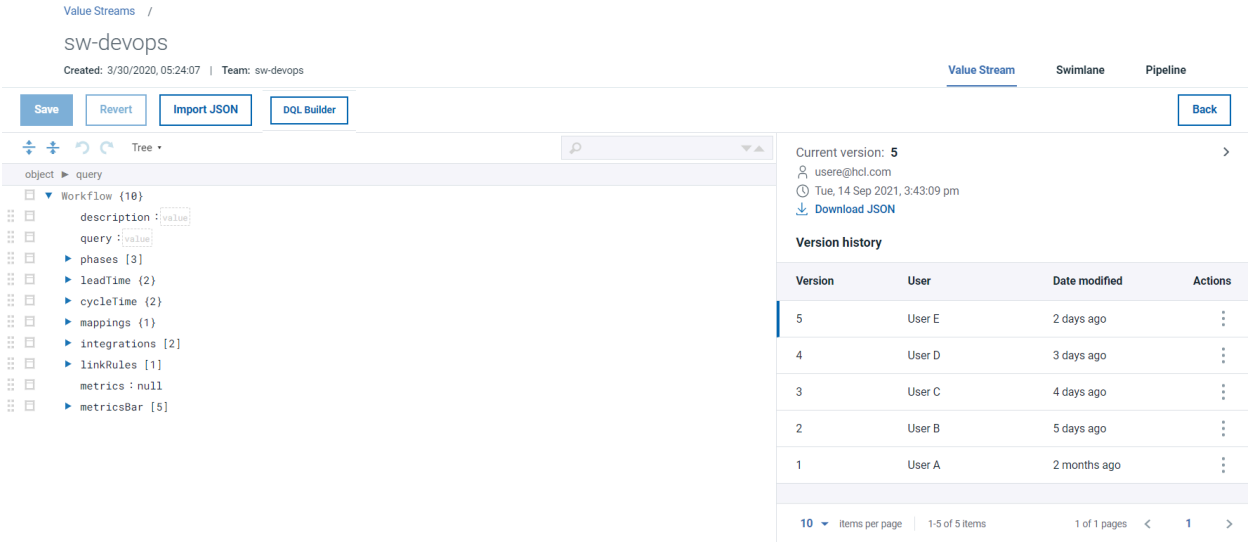
1. On the **Value Streams** page, click the required value stream.
2. Click and select **Edit value stream map** to open the VSM JSON code editor.

   Note: The JSON code editor is the default view mode when you open the editor.

3. On the right of the icon above the code editor, click the **Code** on page 3067 or **Tree** view drop-down and select **Tree**.

   The JSON tree editor opens below the drop-down as shown in the figure below.

   Figure 16. JSON tree view



   Note: In the **Version history** table on the right side of the display, the current version of the JSON shown on the left side of the display is indicated by a vertical blue bar next to the **Version** field in the table and is the file displayed and open in the editor.

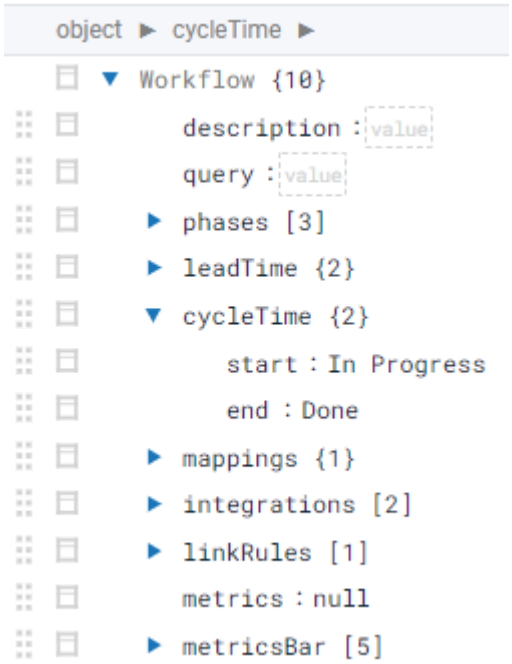
   Note: The icons on the left side of the display above the **Select a node** breadcrumb bar include the following:

   - Click to expand all fields in the tree view.
   - Click to collapse all fields in the tree view.
   - Click to undo last action in the JSON tree view.
   - Click to redo last action in JSON tree view.
   - On the right of the icon above the tree editor is the **Code** or **Tree** view drop-down selector to choose between a plain text editor or a user-friendly editor respectively.
   - On the right of the drop-down selector toward the center of the display is the search bar where you can query for fields and values in the JSON tree. On the right of the search bar are the next result and

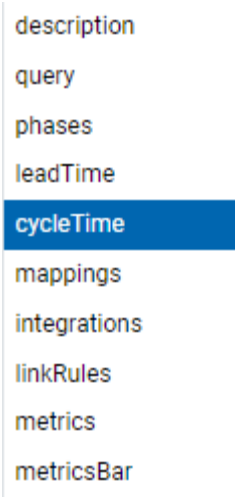
previous result icons shown as and respectively. Queried search items found in the tree editor are highlighted for identification.

- When you click fields in the tree editor, a breadcrumb is displayed in the **Select a node** section based on the selected item in the tree view. For example, if you click **cycleTime** in the tree editor, the breadcrumb is shown in the following figure followed by details:

Figure 17. Select a node breadcrumb



  - On the right side of **object**, click ▶ to display the **object** hierarchy drop-down list shown in the following figure:



Note: Since **cycleTime** was selected, it is highlighted in the drop-down list.

- Click **cycleTime** in the **Select a node** breadcrumb bar and that field is highlighted with a yellow rectangle in the editor.
- On the right side of **cycleTime**, click to display the **cycleTime** hierarchy drop-down list shown in the following figure:

Note: The icons in the tree editor below the **Select a node** breadcrumb bar include the following;

- Click to drag and move fields to different locations within the editor.
- Click to open the actions menu for fields in the tree view. Based on the type of field you click on in the editor, the menus and sub-menus available are detailed in the table below.

Table 1151. Actions menu items

Menu	Sub-menu
Type	**Auto**, **Array**, **Object**, and **String**
Insert	**Auto**, **Array**, **Object**, and **String**
Append	**Auto**, **Array**, **Object**, and **String**
Extract	N/A
Duplicate	N/A
Remove	N/A

Note: Details for items in the table above are located at the top of this page.

- On the right side of , icons are displayed that include the following:
  - Click to expand fields in the tree view.
  - Click to collapse fields in the tree view.

4. To import, download, view, and edit a JSON file in the editor, perform either of the following:

- Click **Import JSON**, select the JSON file, and click the **Open** button.

  Note: The file is opened and displayed in the editor.

- On the right side of the display above the **Version history** table, click **Download JSON** to download the latest JSON file revision shown in the **Current version** field above the creator.

  Note: Above the **Version history** table, you can view detailed information for the latest JSON file with fields including **Current version**, creator, and file creation day, date, year, and time.

- In the **Version history** table with fields including **Version**, **User**, **Date modified**, and **Actions**, locate the required JSON file version and in the **Actions** field, click ⋮ and select either **View code** or **Download** according to your requirement.

  If you want to use a previous version JSON file as the current version, perform the following:

  - In the **Version history** table, click ⋮ for the required previous version and select **Download**.
  - Click **Import JSON** and navigate to the downloaded file.
  - Click **Open**.
  - Click **Save**.

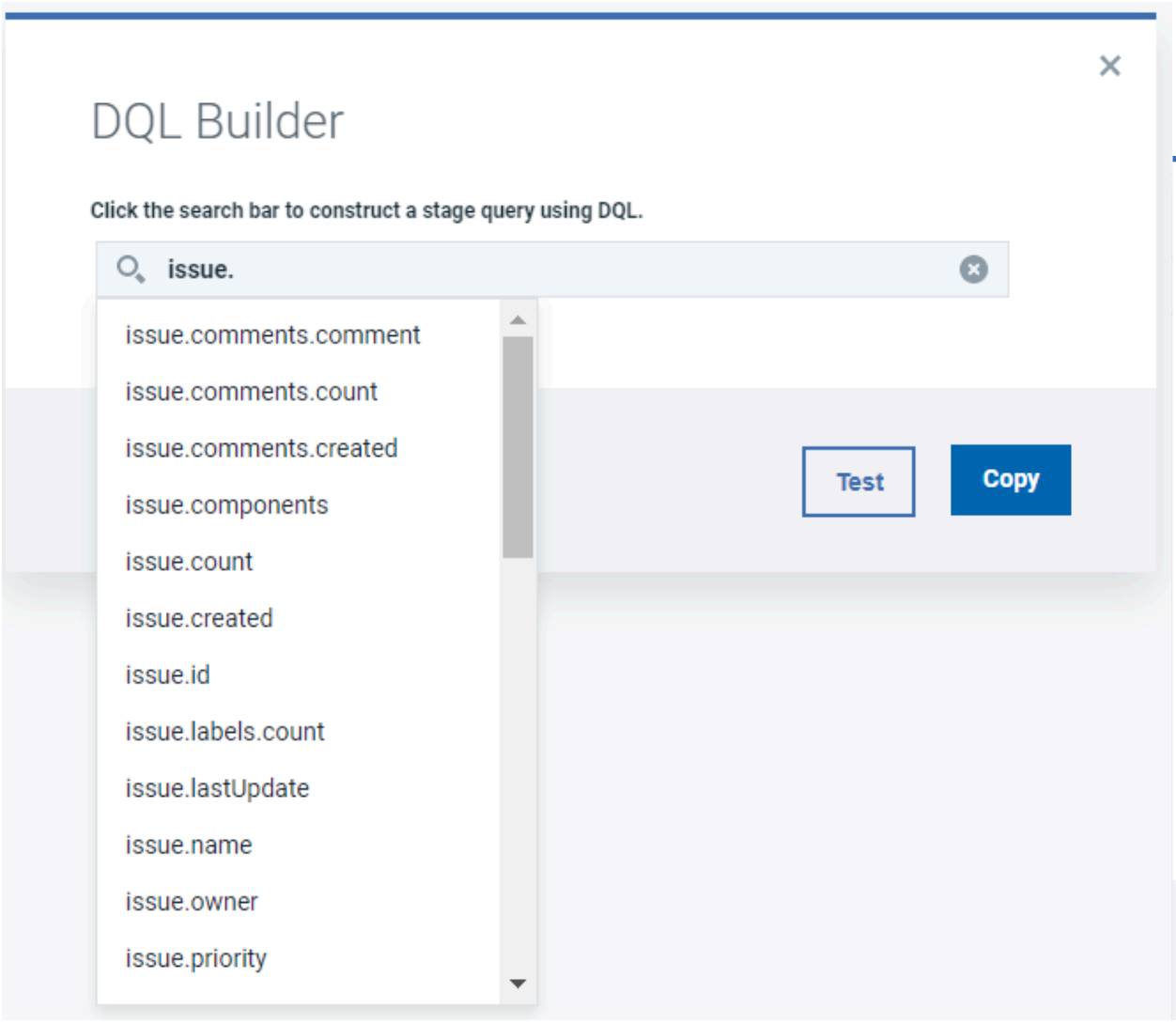
    Note: After a JSON file is successfully saved, you cannot revert the changes that were made prior to saving the file. If you want to use a previous version JSON file as the current version, use the aforementioned procedure.

  At the bottom of the **Version history** table, you can perform and view the following:

  - In the **items per page** drop-down, you can select **10**, **20**, **50**, or **100** as the number of items to display per page in the **Version history** table.
  - View number of items listed in the table versus total number of items. For example: 1-3 of 3 items.
  - View page number versus total number of pages.
  - View the page number currently displayed in the **Version history** table and click **<** or **>** to go to the previous or next page respectively.

5. To create a query, click **DQL Builder** and then **DQL Builder** window appears. Type the query in search field and select the required one from the list to create query. After creating query, click **Test** to view results for the

selected query. Click **copy** to copy the created query. You can use the copied query in JSON code or tree editor.



6. On the left side of the display in the file editor, perform the required editing in the file and click **Save** to capture your changes.

Note: If you make any changes before saving the file, you can click **Revert** to undo all your changes.

Result

When you click **Save**, a notification is displayed containing one of the following messages:

- If all the changes you made to the JSON file are validated when saved, a **Success** notification with a vertical green bar is displayed upon completion.
- If an error is associated with the file, a notification with a vertical red bar is displayed with either of the following information:
  - The source of error.
  - Selected file type is not supported.

7. Overwrite protection provides visibility to users that attempt to simultaneously edit the same workflow. The following example details the procedure for two users simultaneously attempting changes to the same file. User 1 creates and saves version 78 by editing version 77 and User 2 edits version 77 and attempts to save the file with the result of displaying the **OVERWRITE PROTECTION Confirm changes? Users have made 1 new version since loading this page. Do you want to overwrite the latest changes?** modal prompting to click either the **View latest changes** or **Confirm** buttons detailed in the following table:

Button	Description
View latest changes	Modal closes and refreshes the **Version history** table with new saved version 78 and the text editor displays the changes made by User 2 for this example.
Confirm	Saves the changes to version 77 and selects the same version in the **Version history** table for this example.

What to do next

Documentation for the VSM JSON code editor is found here on page 3067.

Integrating tools into value streams

Create value stream integrations by defining integration objects to the `value-stream-name.json` file.

Before you begin

Modify the value stream with the phases, stages, and queries required for your project.

About this task

To define an integration, select the value stream and complete the following steps:

1. On the **Value stream** details page, click and select **Download value stream map**. The `value_stream_name-vsm.json` is downloaded to the specified location.
2. Using a text editor, define an integration object in the `value_stream_name-vsm.json` file.
   The following code example illustrates a Bitbucket™ integration:

```
{
 "type": "ucv-ext-bitbucket-server",
 "tenant_id": "[ID]",
 "name": "my-bitbucket",
 "disabled": false,
 "properties": {
   "baseApiUrl": "[Base URL ex: http://10.134.116.110:7990/rest/api/1.0]",
   "username": "admin",
   "password": "[Basic user password]",
   "projectName": "[Full project name ex: JKE Mortgage Application]",
   "projectKey": "[Project key ex: JMA]",
   "repositoryName": "[app repository name ex: web-application]",
```

```
        "repositorySlug": "[app repository name ex: web-application]"
      }
    }
```

3. **Optional:** Define queries for the stages where data is required from the integrated tool.
   For example, the following query defined for the **In Progress** stage matches open pull requests from source control systems:

```
{
     "name": "Development",
     "description": null,
     "stages": [
       {
         "name": "In Progress",
         "query": "pr.status="open",
         "description": null,
         "wipLimit": null,
         "gates": null
       },
```

4. On the **Value stream** details page, click , select **Edit value stream map > Import JSON**, and then select the JSON file that you modified.
   For more information about creating value stream integrations, see About value stream integrations on page 3078.

Results

Objects from the integrated tool appear in stages of the value stream, and the integration is listed on the **Integrations** page in the web UI.

What to do next

Increase usability by linking integrated tools on page 3066. Linked value stream dots provide information for the linked tools.

About value stream integrations

Integrate external tools into your value streams.

Value stream integrations are specified in the `value_stream_name-vsm.json` file. Each value stream has a unique `value_stream_name-vsm.json` file. You can add any number of integration objects to the JSON file.

You can add integrations to value streams by the following two methods:

1. Create an integration on the Integrations page on page 1881, and then insert a reference to it in the `value_stream_name-vsm.json` file. For example, the following JSON code adds an integration named "DevOps Team" into the value stream, `"name": "DevOps team"`.
2. Define all properties for the integration in the `value_stream_name-vsm.json` file. When you configure an integration with this method, the integration is displayed on the **Integrations** page.

Properties common to all value stream integrations

While all plugins have unique properties, the plugins that you can integrate into a value stream share some common properties. The properties identify the integration and determine if the integration is enabled or disabled.

The properties common to all value stream integrations are detailed in the table below.

Table 1152. Properties common to all integrations

Property	Description
image	The plugin version.
type	The plugin type identifier. See the list of supported plugin types later in this topic.
tenant_id	Tenant identifier. The value is generated by Measure.
name	Integration name. The name should be unique.
disabled	A flag that is used to disable the integration. The default value is `null`, which means the integration is enabled. To disable the integration, change the value to `true`. You can also set this value with the **Disable/Enable** button on the **Integrations** page.
properties	The `properties` object defines properties that are specific to the plugin. See the documentation for the tool that you want to integrate on page 1874.

For information regarding parameters unique to a specific integration, see Introducing plugins on page 1874.

The following code block shows a typical Jira-type integration defined in the `value_stream_name-vsm.json` file:

```
"integrations": [
    {
      "type": "ucv-ext-jira",
      "tenant_id": "tenantid",
      "name": "Jira_Plugin ",
      "properties": {
        "baseUrl": "jira_server_url",
        "username": "jira_server_user_name",
        "password": "jira_server_password",
        "consumer_key": null,
        "consumer_secret": null,
        "access_token": null,
        "access_token_secret": null,
        "jiraProjects": [
          "jira_project_name"
        ],
        "proxyServer": "proxy_server_url",
        "proxyUsername": "proxy_server_user_name",
        "proxyPassword": "proxy_server_password"
```

```
    }
  }  ]
```

Linking rules

Link value stream tools to one another.

Link rules define how integrated tools relate to one another. When tools are linked, objects that appear in a value stream provide information and links to the associated tools. You might link an issue tracking system, for instance, to a source control tool.

To define a link rule, specify a field in the linked-from tool that you want to associate with a field in the linked-to tool, and a regular expression that defines the matching pattern.

The following example illustrates how to link a Git integration with a Pivotal integration:

- The `"fromIntegrationName": "my-github"` property identifies the Git integration defined in the value stream.
- The `"fromField": "pr.name"` or `"fromField": "pr.description"` property links the Git pull request name to the linked-to tool.
- The `"toIntegrationName": "my-pivotal"` property identifies the Pivotal integration defined in the value stream.
- The `"toField": "issue.id"` property links the Pivotal issue ID to the linked-from tool.

  Note: The default format of the Jira `id` is `Board-Number`. For example, if the Jira `issue.id` is `JIR-1` and it was linking to a PR via the aforementioned issue by `pr.name`, the PR name *must* include the text `JIR-1`.

- The `"pattern": "[[A-Z]+-[0-9]+}"` property defines the regular expression that determines if potential links match.

  Note: The `pattern` property uses regex to extract the desired string from a field. PR names often include a description of the PR along with the id of the corresponding Jira issue. When this occurs, it is necessary to specify what value from the `fromField` should be used for linking. If the value extracted from the `fromField` is not an *exact* match with the `toField`, the integrations will *not* be linked.

```
"linkRules": [
  {
    "toIntegrationName": "my-pivotal",
    "toField": "issue.id",
    "pattern": "[[A-Z]+-[0-9]+]",
    "fromIntegrationName": "my-github",
    "fromField": "pr.name",
  },
 ]
```

When links are established, you can access information about the linked tools from the affected dots in the value stream. In general, you can define linking rules for any integrated tools, and define multiple linking rules in a single value stream.

Note: To add linked data to a value stream, you must define linking rules for your integrated tools.

Table 1153. Link rule properties

Property	Description
toIntegrationName	Name of the integration to which you are linking.
toField	Field in the `toIntegrationName` integration that you are linking.
pattern	Regular expression that determines if the linking fields match. For example, `[[A-Z]+-[0-9]+]`.
fromIntegrationName	Name of the integration from which you are linking.
fromField	Field in the `fromIntegrationName` integration that you are linking.

Viewing value streams

Viewing value streams using list and card view layouts.

About this task

After you create one or more value streams, you can view them by clicking **Value Streams > All Value Streams** in either the list or card view layouts. In both the layouts, click icon to view the corresponding pipeline page. You can also add metrics.

1. To view value streams in the list view layout and add metrics, complete the following steps:
    a. On the Measure dashboard, click **Value Streams > All value streams**.
    b. Click to display the list view of value streams. The default fields are **Name**, **Lead Time**, and **Cycle Time**. You can click on the name of any value stream in the list view to display it in the **Value stream** tab. You can also sort the **Name** field in either ascending or descending order by clicking on the triangle in the right side of the field.
    c. Click to display the **Add Metrics** window.
    d. Click to add the required metric *or* click to expand the required metric that is added to the **Metric bar**.

    Note: For metrics having an expand and collapse button, you can define the metric in the following step. For metrics having only an add button, **Metric added!** is displayed when you click the button.

    e. Define the metric by completing the following steps:

i. In the **Metric Title** field, enter a name for the metric.
ii. In the **Relative time** field, select a time range. Data within the time range are calculated.
iii. In the **Filtered values** field, check the required filter based on attributes of add metric.

Note: The **Filtered values** field is only available for certain metrics.

iv. In the **Warn when metric data is missing from applications** field, click either the **Any application** or **All applications** radio button.

Note: The **Warn when metric data is missing from applications** field is only available for certain metrics.

v. Click the **Add** button to display the metric in the value stream tabular list view.

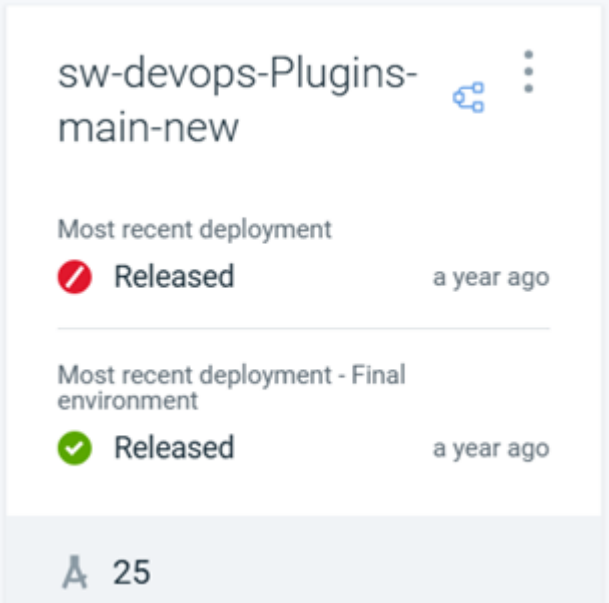
Note: There is no limit on the number of metrics that can be added in a value stream.

vi. To edit an added metric with other metric column options in list view layout, in the required metric field, hover and click ••• toward the right side of the metric column and select one of the following menu items:
- **Edit metric** to display the **Edit Metric** window to modify the metric parameters available for the item. After making the required selections, click the **Update** button to save your changes.
- **Only show value streams missing this data** for the related metric.
- **Remove metric** to delete the metric.

Result

For added metrics in list view layout, there may be color icons appearing in the metric columns. Hover over the icon to display a pop-up with specific details about the metric.

2. To view value streams in the card view layout, complete the following steps:
   a. On the Measure dashboard, click **Value Streams > All Value Streams**.
   b. Click to display the card view of value streams. An example card view for a value stream is shown in the following figure.



Information and menu options on the value stream card view includes the following explained in reference to the above figure:

- **Test** is the name of the value stream. You can click on the name of any value stream in the card view to display it in the **Value stream** tab.
- **Most recent deployment** is **PROD** deployed **2 months ago**. The red slash icon indicates a failed deployment.
- **Most recent deployment - Final environment** is **PROD** deployed **2 months ago**. The green check icon indicates a successful deployment.
- 2 indicates there are two pipeline applications for the value stream.
- Click and select one of the following menu items:
  - **Edit** to display the **Edit Value Stream** window to modify the **Name**, **Description**, or **Team**. Click the **Save** button to save your changes.
  - **Delete** to remove the value stream card.

What to do next

Next, filter value streams by creating queries with the DevOps query language on page 3117. Learn about the information presented by value streams. on page 3103

Value stream metrics

Assess project quality, progress, and performance with the Measure metric data tools.

The ability to measure project status is a core requirement. Measure provides performance metrics for every project area, from initial concept to final delivery. All project areas can be accessed for efficiency and quality. You can add metric performance data to value stream, Insights dashboards, and applications.

To add performance metrics to your value stream, query the data that you want to analyze, and then apply metric performance chats to the data. The data can come from integrations with testing tools, performance monitors, and security scanning applications, to name just a few. You can display performance metrics in charts and graphs on page 3084 on the value stream **Metrics** bar.

You can add custom metrics to your Insights dashboard on page 3091, or use the prebuilt charts.

Displaying the metrics on the value stream

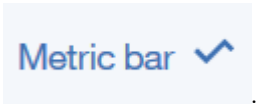
You can enable the **Metric bar** to display the metrics on the value stream.

You can provide metrics data to value streams and display the data in charts on the **Metric bar**. To enable the **Metric bar**, add `metricsBar` definitions to the `value_stream_name-vsm.json` file attached to the value stream.

To display the metric bar data to the `value_stream_name-vsm.json` file, complete the following steps:

1. On the **Value Streams** page, select a value stream.

   Value stream details page is displayed.

2. Click and select **Edit value stream map**.
3. Add `metricsBar` definition to the `value_stream_name-vsm.json` file. The properties are defined later in this topic.
4. Click **Save**.
5. On the **Value stream** details page, click 

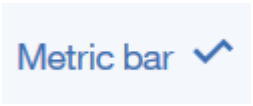
.

   Note: The metric bar displays **Lead Time**, **Cycle Time**, **Dev Cycle Time**, **Distribution**, and **Throughput** metrics by default.

After enabling the metric bar, you can click to add metrics to the value stream.

Adding metrics to the metric bar

To add metrics to the **Metric bar**, complete the following steps:

1. On the **Value Streams** details page, click 

to display the metric bar.
2. Click to display the **Add Metrics** window.
3. Click to add the required metric *or* click to expand the required metric that is added to the **Metric bar**.

Note: For metrics having an expand and collapse button, you can define the metric in the following step. For metrics having only an add button, **Metric added!** is displayed when you click the button.

4. Define the metric by completing the following steps:
    a. In the **Metric Title** field, enter a name for the metric.
    b. In the **Relative time** field, select a time range. Data within the time range are calculated.
5. Click **Add** to insert the metric into the **Metric bar**.
6. The **Metric added!** pop-up is displayed next to the icon.
7. To remove a metric, hover over the metric, click and then select **Remove metric**.

Note: In addition to lead time, cycle time and dev cycle time, you can select up to six metrics for a total of eight metrics able for display at one time on the metric bar. Also, when you click on either the **Lead Time**, **Cycle Time**, or **DevCycle Time** on the **Metric bar**, the **Lead time distribution**, **Cycle time distribution**, or **Dev time distribution** graph window is respectively displayed.

Metric bar JSON parameters

Metric bar parameters are defined in the `metricsBar` area of the `value_stream_name-vsm.json` file.

Table 1154. Efficiency metrics

Efficiency Metric	Description
`relativeTime`	Date range for data calculation. Valid values include `HOUR24`, `DAY7`, `DAY30`, `DAY90`, `MONTH6`, `YEAR1`, and `ALL`.
`metricDefinitionId`	Metric definition to apply. Any metric definition defined to Measure can be referenced.
`name`	Metric name
`displayType`	Determines how data is rendered. Valid values are: `bar`, `percent`.
`calculationType`	Determines how the data is aggregated. Valid values include `mostRecent` and `trendMostRecent`. The `mostRecent` value displays the most recent run compared to the average. The `trendMostRecent` adds a trend breakdown to the most recent data.

The following example adds a bar chart for work items over the past 30 days.

```
"metricsBar": [
   {
     "relativeTime": "DAY30",
     "metricDefinitionId": "WORK_ITEM_TYPE_COUNT",
     "name": "Work Item Type Count",
     "displayType": "bar",
```

```
      "calculationType": "trendMostRecent"
    }
  ]
```

Metrics definitions

On the **Value Streams** page metric definitions are detailed in the following tables.

Delivery flow-type charts

Table 1155. Delivery flow-type charts

Chart	Description
Lead Time	The time required for a dot to go from business acceptance to value realization. This includes both engineering time and non-value-added waiting time between sub-processes.
Lead Time Key Performance Indicator (KPI)	The Lead Time KPI measures the efficiency of a dot to reach the defined target. You can define the target using the status rule. To define target for lead time KPI, see here on page 3173.
Cycle Time	This is similar to lead time, except cycle time is measured from when the development team begins work to when that work reaches the customer.
Cycle Time Key Performance Indicator (KPI)	The Cycle Time KPI measures the efficiency of a dot to reach the defined target. You can define the target using the status rule. To define target for cycle time KPI, see here on page 3172.
Dev Cycle Time	This is similar to cycle time, except dev-cycle time is measured from when the development team begins work to when that work reaches end of the development (before the production stage).
Build Counts	Number of builds by type in a specified time period.
Deployment Counts	Number of deploys by status in a specified time period. The metric can be set to process names, environment, result, type, or application. The default metric is result. The data can scoped by data set, environment, process, result, or version.
Load	The number of work items active or waiting in a value stream at a given time. Load measures utilization capabilities of value streams related to productivity in

Table 1155. Delivery flow-type charts (continued)

Chart	Description
	the process flow. "Active" refers to the stages from the lead-time start until the lead-time end.
Throughput	The rate of work items completed during a period of time. Improving throughput can result in better responsiveness to customer requirements and may yield lead time reductions for value streams.
Agile Churn	The ratio of the sum of the values of story points added and removed to the value of the planned story points of a sprint. It calculates the percentage of the change in planned scope for any sprint.
Distribution	The proportion of different types of work items over time. This provides teams visibility into the type of work being completed (features, defects, tasks, and so forth.)
Flow Efficiency	The ratio of actual development time to cycle time. It is expressed as a percentage, which helps teams to understand their performance.
P.R.E Risk	Probabilistic Risk Estimator (P.R.E) Risk provides estimated probability to miss current sprint goal.
P.R.E Deadline	P.R.E Deadline provides estimated complete date for the current sprint.

Table 1156. DORA charts

Chart	Description
Deployment Frequency	The number deployments into production environment.
Lead Time to Change	The time required for a dot to go from commit to production deployment.
Time To Restore Service (MTTR)	The time taken to resolve the incident.
Change Failure Rate (CFR)	The ratio of the number of failed changes to the total number of changes deployed within a specified time period. It is expressed as a percentage, helping teams understand the rate at which their deployments are successful or failed.

For uncommon metric names between the **Value Streams** and **Insights** pages associated with delivery charts, refer to the following equivalence table for details.

Metric name equivalences

Table 1157. Metric name equivalences

Value Streams	Insights
Build Frequency (number per time)	**Build Counts** (Number of builds by type in a specified time period.)
Average Load	**Load**
Bottlleneck A constriction in local throughput that limits the global throughput of a system. Bottlenecks include a set of events or conditions that occur in the value stream that you monitor in the process. Refer to the bottleneck detection page on page 3098 for further details. Note: *Bottleneck is not a metric.* This indicator can be added and viewed only on the metrics bar and the value stream table view.	N/A

Quality-type charts

Table 1158. Quality-type charts

Chart	Description
Coverage By Branch	Percent of branches with associated tests executed during testing.
Coverage By Function	Percent of program functions executed during testing.
Coverage By Line	Percent of lines of code executed during testing.
Functional Tests	Number of functional tests by status in a specified period of time.
Static Code Analysis	Number of issues by severity in a specified period of time.
Unit Tests	Number of tests by status in a specified period of time.
API Tests	Number of tests by status in a specified period of time.

Risk-type charts

Table 1159. Risk-type charts

Chart	Description
Container Vulnerabilities	Number of container vulnerabilities by severity in a specified period of time. The metric can be set to process names, environment, result, type, application, tag, or team. The

Table 1159. Risk-type charts (continued)

Chart	Description
	default metric is result. The data can scoped by data set, process, or version.
Application Vulnerabilities	Number of application vulnerabilities by severity in a specified period of time.

Customizing metric definitions for Lead Time, Cycle Time, and Dev Cycle Time

To customize the metric definitions for calculating the **Lead Time**, **Cycle Time**, and **Dev Cycle Time** metrics, perform the following steps:

1. Hover over the metric, click ••• and then select **Edit**.

   The **Edit Metric** dialog is displayed.

2. Select one or more input stages from the drop-down list of **Start Stage** and **End Stage**.

   The default metric definitions are as follows:

Metric	Start Stage	End Stage
Lead Time	Backlog	Completed
Cycle Time	In Progress	Completed
Dev Cycle Time	In Review	Completed

3. Click **Update**.

   The updated metric definitions are saved in the vsm.json file, and the static metric graphs are updated for **Lead Time**, **Cycle Time**, and **Dev Cycle Time** in Insights dashboard.

Customizing metric definitions for Lead Time, Cycle Time, and Dev Cycle Time

You can now select multiple start and end stage for a metric, enabling a more accurate representation of the actual workflow across teams. Because the definition of work start and end times can vary depending on the tools, processes, and delivery practices used. This provides teams and stakeholders with clearer insights into the actual flow of work, supporting better decision-making, increased transparency, and continuous improvement adaptability for the team.

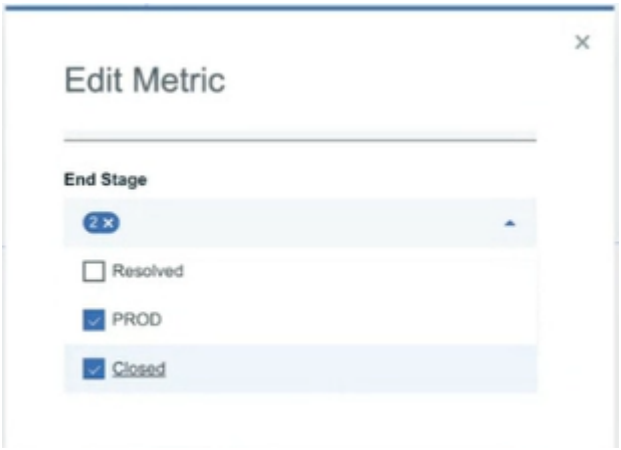
The metric is calculated as the median duration between all selected start and end stages. This ensures the measurement reflects the entire workflow and accurately captures the time spent across all phases.

To customize the metric definitions for calculating the **Lead Time**, **Cycle Time**, and **Dev Cycle Time** metrics, perform the following steps:

1. Hover over the metric, click ••• and then select **Edit**.

   The **Edit Metric** dialog is displayed.

2. Select one or more stages from the drop-down list of **Start Stage** and **End Stage** for the metric.



3. Click **Update**.

   The updated metric definitions are saved in the vsm.json file, and the static metric graphs are updated for **Lead Time**, **Cycle Time**, and **Dev Cycle Time** in Insights dashboard.

Example: Calculation of Cycle time when multiple stages selected for end stage

The following setup describes how the Cycle time is calculated when multiple End stages are selected. In this example, Start stage is selected as **Active**, whereas **Production** and **Closed** are selected as End stage.

Work item	Start Stage	Start date	End Stage	End date	Duration
A	Active	June 1	Closed	June 7	6 days
B	Active	June 3	Production	June 10	7 days
C	Active	June 3	Production	June 7	4 days
D	Active	June 2	Production	June 5	3 days
E	Active	June 7	Production	June 12	5 days
F	Active	June 5	Production	June 10	5 days

Cycle time calculation: The distribution of the cycle times of all work items completed during the last 30 days. The calculated durations from the above table are sorted in ascending order. For example, if the duration values are [6, 7, 4, 3, 5, 5], they are reordered as [3, 4, 5, 5, 6, 7]. The **Median Cycle time** over this period is **5 days**, with 75 percent of items reaching completion within approximately 6.25 days.

Implementing custom metrics

Provide custom metric data to your value streams and dashboards.

Custom metric definitions can be defined in the `value_stream.json` file attached to a value stream. You can add graphs or charts derived from the data to dashboards, the value stream **Metrics** bar, or the list view on the **Value Streams** page.

Note: Until new users are assigned to teams, they are automatically assigned to the Default team in the Viewer and Participant roles. Users in these roles can view objects, such as releases and value streams, but they cannot create or edit them. Additionally, users in these roles can generate user access tokens and access API endpoints with GET requests on page 1820. Administrators grant users elevated permissions when they assign them to roles such as Lead Developer or Release Manager.

The following steps describe how to add custom metric data to Measure:

1. Using the properties defined later in this topic, define a custom metric definition.
2. Upload the metric definition to Measure.
3. Send data to Measure that conforms to the metric definition.
4. Add charts based on the custom data to value streams and dashboards.

Metric definition properties

Metric data properties are described in the following table:

Table 1160. Value stream metrics properties

Property	Description
id	This string-type property is used to link metric records to the metric definition. Required.
name	This string-type property identifies the metric definition. The name appears in the **Add charts** area of the Dashboards page. Required.
category	This string-type property defines the category to which the metric definition belongs. The categories are listed in the **Add charts** area of the Dashboards page. Possible values include: `quality`, `deliveryFlow`, `risk`, `team`, and `effort`. Required.
tenantId	This string-type property is generated by Measure and is used by the product to identify your installation. You can find your tenant ID by downloading a value stream JSON file. Required.
valueType	This string-type property indicates the expected value of the metric data. Possible values include: `percent`, `countset`, and `count`.

Table 1160. Value stream metrics properties (continued)

Property	Description
chartDefaults	This JSON object sets default options for the chart associated with the metric definition. The object properties are: `groupBy` (string), `chartType` (bar or line), `barMode` (stack or group), and `threshold` (number).
aggregation	This JSON object sets the default aggregation logic.
createdBy	Name or ID of the metric definition creator. String
description	Metric definition description. String.

Example to add metric definition

The following example shows the curl command for a metric definition. Measure uses the `"Authorization: UserAccessKey"` property to authenticate client access. You can create an user access token on the **User Access Keys** page, **Settings > My Profile**.

```
curl -X POST "https://localhost/api/v1/metricDefinition" \
 -H "accept: application/json" \
 -H "Authorization: UserAccessKey my_user_access_key" \
 -H "Content-Type: application/json" \
 -k \
 -d '{
       "id": "MY_CUSTOM_DEFINITION",
       "name":"My Custom Definition",
       "category": "quality",
       "tenantId":"my_Velocity_TenantID",
       "chartDefaults":{
         "groupBy":"result",
         "chartType":"bar",
         "barMode": "stack"
       },
       "aggregation":{
         "operation":"sum"
       },
       "createdBy": "A. Coder",
       "description": "New custom metric"
     }'
```

Example to update metric definition

The following example shows the curl command to update an existing metric definition. Measure uses the `"Authorization: UserAccessKey"` property to authenticate client access. You can create an user access token on the **User Access Keys** page, **Settings > My Profile**.

```
curl -X POST "https://localhost/api/v1/metricDefinition" \
 -H "accept: application/json" \
 -H "Authorization: UserAccessKey my_user_access_key" \
 -H "Content-Type: application/json" \
 -k \
 -d '{
```

```
      "operationName":null,
      "variables":{},
      "query":"mutation {
       updateMetricDefinition(input: {
       id:\"TUTORIAL_DEFINITION\",
       tenantId: \"my_Velocity_TenantID\",
       updates:{
       name: \"Tutorial Definition Changed 8\",
       category: \"quality\",
       chartDefaults:{
         groupBy: \"result\",
          chartType: \"bar\",
          "barMode": \"stack\"
         },
         aggregation: {
         operation: \"sum\"
         }
        }
       }){
       id
       name
       }
       }"
       }'
```

Example to delete metric definition

The following example shows the curl command to delete an existing metric definition. Measure uses the `"Authorization: UserAccessKey"` property to authenticate client access. You can create an user access token on the **User Access Keys** page, **Settings > My Profile**.

```
curl -X POST "https://localhost/api/v1/metricDefinition" \

 -H 'accept: application/json' \

 -H "Authorization: UserAccessKey my_user_access_token" \

 -H 'content-type: application/json' \

 -d '{
      "operationName":null,
       "variables":{},
        "query":"mutation {
         removeMetricDefinition(input:{
         id: \"TUTORIAL_DEFINITION\",
         tenantId: \"my_Velocity_TenantID\"
         }
         )
        {
        id
        name
        }
        }"
        }'
```

DevOps Research and Assessment (DORA) metrics

You can use DORA metrics to evaluate the performance of the team.

The following DORA metrics are available in Measure

- Lead Time for Changes
- Deployment Frequency
- Time To Restore Service (MTTR)

Lead Time for Changes

The lead time for changes is the quantity of time it takes a dot to progress from commit to production deployment. The metrics calculation is based on the date of the commit and date of the deployment into production. The lead time for changes metrics is a default metric added to all available value streams metrics bar.

Complete the following steps to add lead time for changes metrics to Insights:

1. On the **Value streams** page, click **All value Streams**.
2. Click a value stream.
3. Click and select **Edit value stream map** to modify the JSON file in the code on page 3067 or tree on page 3071 view editors.

   You can also click and select **Download value stream map**.

   The value_stream_name-vsm.json file is downloaded to the specified location.
4. Update the leadTimeChange metric in the JSON file as shown in the following example:

   The following example is based on the assumption that your commit is queried in the **In Review** stage and the deployment to production is queried in the **PROD** stage of your value stream.

```
"leadTimeChange": {
    "start": In Review,
    "end": PROD
  }
```

Deployment Frequency

Deployment Frequency Counts assess the number of deployments into production over a time. The metrics calculation is based on the deployment start date and end date.

Complete the following steps to add Deployment Frequency metrics to value streams.

1. On the **Value streams** page, click **All value streams**.
2. Click a value stream.
3. On the **Value stream** details page, click to add the required metrics.
4. In the **Add metrics** window, perform any one of the following steps to complete the addition of the production **Deployment Counts** metric:
    a. Click to add a production **Deployment Counts** metric with the default values for parameter fields. See the following table for parameters, descriptions, and default values.

Parameter	Description	Default value
Metric title	The title of the metric to display in value steams metrics bar.	Production deployment counts
Relative time	The time to use for the metric calculation	30 days or a custom value
Stages	The staging environment of the pipeline	Measure takes the last viewed staging environment

    b. Click the down arrow to enter the details for the metric. See the following table for parameters and descriptions:

Parameter	Description
Metric title	The title of the metric to display in value steams metrics bar.
Relative time	The time to use for the metric calculation
Stages	The staging environment of the pipeline

Complete the following steps to add Deployment Frequency metrics to Insights:

1. On the **Insights** page, in the **All dashboards** tab, click any dashboard.
2. On the selected dashboard page, click **Add charts**.
3. In the **Add charts** section, click the right angled arrow for **DORA** to see the list of metrics.
4. Click **Add+** for **Deployment Frequency** to add the metric.
5. The chart area displays the data for **Deployment Frequency** metric.

Time To Restore Service (MTTR)

Time to restore service is a metric to calculate the time that team members spend to resolve the reported incident. This metric calculation is based on the incident creation date and incident resolution date.

To view the severity levels You must integrate the Jira Srrvice Management plug-in to view the severity levels.

Complete the following steps to add Time To Restore Service metrics to value streams.

1. On the **Value streams** page, click **All value streams**.
2. Click any value stream.
3. On the **Value stream** details page, click to add the required metrics.
4. In the **Add Metrics** window, perform any one of the following steps to add **Time To Restore Service** metric:
   a. Click to add **Time To Restore Service** metric with the default values for parameter fields. See the following table for parameters, descriptions, and default values.

Parameter	Description	Default value
Metric title	The title of the metric to display in value steams metrics bar.	Time to Restore Service
Relative time	The time to use for the metric calculation	30 days or a custom value
Aggregation interval	The interval for data aggregation	Days
Severity	The incident severity levels are a measure of the impact an incident has on the business.	SEV1, SEV2, SEV3, SEV4, and SEV5

   b. Click the down arrow to enter the details for the metric. See the following table for parameters and descriptions:

Parameter	Description
Metric title	The title of the metric to display in value steams metrics bar.
Relative time	The time to use for the metric calculation
Aggregation interval	The interval for data aggregation
Severity	The incident severity levels are a measure of the impact an incident has on the business.

Complete the following steps to add time to restore service metrics to Insights:

1. On the **Insights** page, in the **All dashboards** tab, click any dashboard.
2. On the selected dashboard page, click **Add charts**.
3. In the **Add charts** section, click the right angled arrow for to see the list of metrics.
4. Click **Add+** for **Time To Restore Service** to add the metric.

The chart area displays the data for **Time To Restore Service** metric.

5. Click **Time To Restore Service** metric to view the severity levels.

Milestone risk estimation

Milestone risk is a metric which measures the probability of a team that may not complete all of the current in-scope work in an active development iteration (sprint). The milestone risk estimator plug-in uses a stochastic machine learning (ML) algorithm to estimate and predict the potential percentage risk of non-completion of all work during an on-going sprint. To calculate this metric, the model uses all historical work items available in the selected value stream to train the model and extracts the required information to make the prediction.

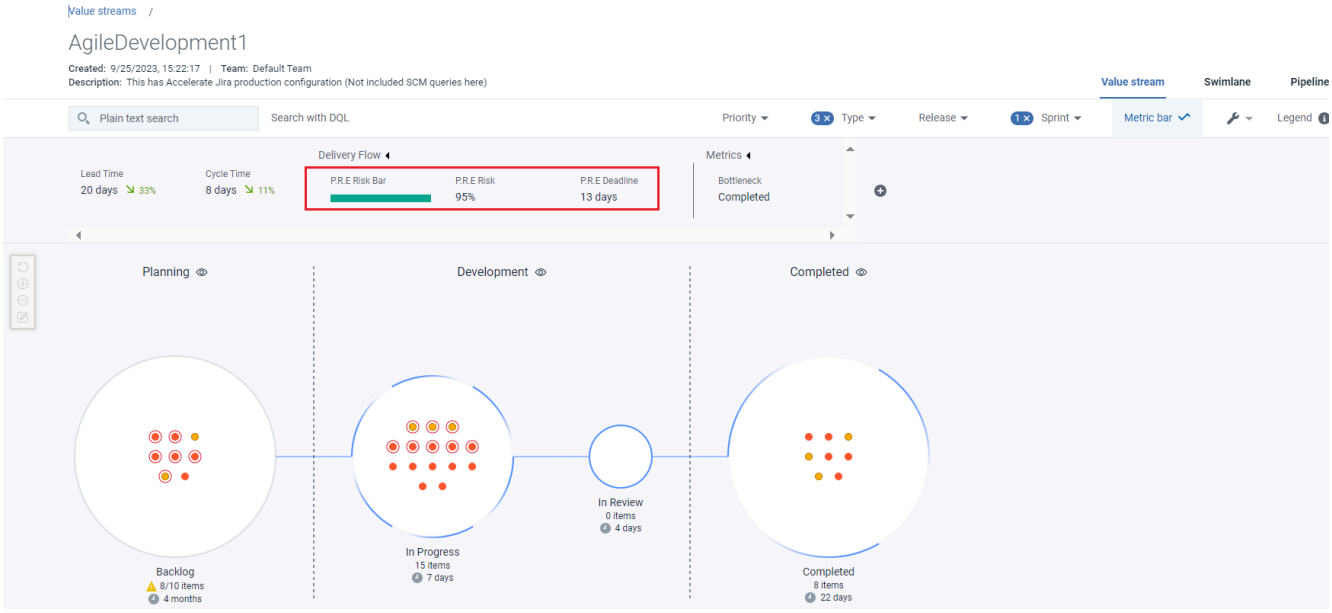
The training data plays a vital role in the process of building, training, and predicting risk. The milestone-risk estimator model uses Cycle time on page 3170 metrics to estimate risk.

The training dataset is updated at regular daily intervals to add the completed work items to improve the efficiency of the risk estimator model. You must select the required value stream when integrating the milestone-risk estimator plug-in in Measure.

The milestone-risk estimator plug-in estimates risk in the current sprint of the value stream and provides estimated results in the form of a Probabilistic Risk Estimator (P.R.E) metrics:

- **P.R.E Risk** displays the evaluated risk as a percentage.
- **P.R.E Deadline** displays an estimated deadline for the sprint.

To view the risk estimation, you must add **P.R.E Risk** and **P.R.E Deadline** metrics under the **Delivery flow** category on the Value streams page. See Displaying the metrics on the value stream on page 3084.



Data restrictions

The restrictions for the milestone-risk estimation model are as follows:

- The risk estimator model does not use stories and epics to train the model or predict the risk. The current model takes in only tasks, sub-tasks, and bugs into consideration for training and predicting risk.
- The risk estimator model considers 7 potential work days a week instead of 5 days.
- The risk estimator model assume serial work from an individual team member. The overlapped time between multiple tickets is divided if a user works on them all and completes them in the same sprint.

  For example, consider a user who works on ticket A from day 1 through day 5, but does not update the ticket to closed until day 10 of the sprint. The user also started working on ticket B from day 6 through day 10, but does not close it until day 15 of the sprint.

Bottleneck detection

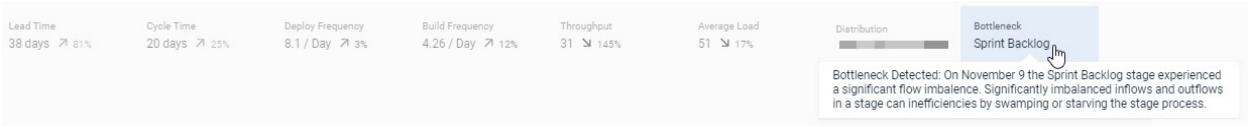
Bottleneck detection monitors, detects, and alerts user of restrictions in a value stream.

Utilizing the power of machine learning and AI, the bottleneck detection algorithm monitors the current and previous state of the value stream to identify which stage the user should focus on first. Bottleneck detection is viewed by either of the following methods:

- On the **Value Streams** page with the required value stream selected, click **Value stream > Metric bar**.
- Click **Value Streams > All Value Streams**. In the tabular view, the **Bottleneck** field will identify the restriction in the value stream.

The bottleneck detection plugin is used to identify a value stream restriction in local throughput that limits the global throughput of a system. The plugin uses work item stage-change data in unison with three algorithms to detect the presence of a specific type of bottleneck in a value stream. Detected bottlenecks are displayed in the metrics bar in the form of the following:

1. Bottleneck location displayed as value stream and stage.
2. Detection message displayed when you hover over the identified bottleneck.
   Figure 18. Bottleneck detection



3. Measure of bottleneck severity level used for display prioritization.

Note: The bottleneck detection parameter is not a value stream metric.

Process

The bottleneck detection plugin runs on a frequency set by the user during the initial configuration. Typically, bottlenecks do not vary continually and a frequency of 24 hours is adequate for most cases.

At runtime, the plugin reads `stageChangeEvents` records via the `stageChangeEventsSince` graphQL API for all value streams defined on an Measure instance. The records for each value stream are then processed through each individual algorithm. Bottleneck detection records are then returned for each value stream. A value stream may have multiple bottleneck detection records created during a given run, or it may have none. If multiple bottlenecks are found in a given run, the Measure UI will only display the highest severity bottleneck while logging the full set of detections in the database.

After the detection process is completed, the identified bottleneck records are written to the `metrics` database via the `uploadMetrics` graphQL API.

Bottleneck algorithms

The plugin uses three algorithms that each detect a unique class of bottlenecks according to different category processes that include the following:

1. Dominant stage bottleneck
2. Batching bottleneck
3. Inflow/outflow asymmetry bottleneck

A **dominant stage bottleneck** occurs when one stage in a value stream has a time-in-stage value greater than or equal to the sum of the time-in-stages of all other stages in that value stream. The dominant stage detection algorithm detects this condition in a value stream and uses the severity value of the dominant stage to non-dominant stages time-in-stages to assess the severity. Each value stream will have only one dominant stage bottleneck detection at a given time.

A **batching bottleneck** occurs when work items exit a given stage in short time intervals separated by large intervals with no associated activity. This algorithm detects batching by calculating stage exit event densities and comparing peak densities to off-peak densities. The severity of the batching bottleneck corresponds to the peak to off-peak density ratio. A value stream may contain multiple stages with batching bottlenecks at a given time.

An **inflow/outflow asymmetry bottleneck** occurs when there is an unusually large difference of work items flowing into and out of a stage on a given day. This algorithms monitors 24 hour periods where the net work item flow for a stage is greater than two standard deviations from the mean net flow, and total stage flow is in the top five percent of the historical stage flow records. The severity of the bottleneck is assessed by the amount value above the net flow detection threshold. A value stream may have multiple stages with inflow/outflow asymmetry bottlenecks at a given time.

Example bottleneck detection output

The following example shows the bottleneck detection output:

```
{
 tenantId : tenantId,
 workflowId : workflowId,
 dataSet : "Bottleneck Detection",
 record : {
               entries : {
                           alertType : "Bottleneck",
                           alertLocation :  bottleneckStageName,
                           alertString : bottleneckMessage,
                           alertLevel : severityMeasure,
                           alertLink : None
                         }
         }
}
```

Bottleneck detection display troubleshooting

For problems and solutions associated with display of the bottleneck detection parameter, refer to the following table:

Table 1161. Bottleneck detection display troubleshooting

Problem	Solution
The plugin instance is not configured or activated.	Verify the plugin instance is enabled and has a valid access key.
For a new or not fully configured value stream (one without valid stage definitions) there will not be sufficient data to detect bottlenecks.	Ensure the value stream definitions are properly configured and associated historical data exists in the system. Preferably at least 30 days of records with a minimum of 10 days.
No bottlenecks are detected. It is possible for a value stream to not have any of the bottleneck conditions which the plugin detects.	Check the plugin logs to verify the plugin is executing successfully.

Metrics FAQs

Metrics frequently asked questions

Example

How do I add to the metric bar? What permission and roles are required to modify the metric bar?

You can select up to eight metrics including lead- and cycle-time along with six more that you choose for your value stream. These metrics are data driven from integrations and value stream data. A user can add metrics by clicking on the **Metric bar** if available and then follow these instructions on page 3084. If it is not available, it means you either do not have permissions or the metric bar already has a maximum of eight metrics displayed.

Note: The

`CreatePipeline` permission is required through either the Lead developer or Pipeline designer roles to add metrics.

What do these metrics mean?

Metric definitions are found here. on page 3084 You can also view metric definitions on the following pages:

- On the **Value stream** details page, click
    1. Metric chart reference on page 3170

  to display the **Add Metrics** window and then hover over the individual metrics underneath the **Delivery flow**, **Quality**, and **Risk** metrics categories to display metrics with definitions in pop-ups.
- On the Insights **Dashboards** page, click **Add charts**. Metric definitions can be viewed by clicking the expand and collapse buttons on the left side of the **Delivery flow**, **Quality**, and **Risk** metrics categories followed by clicking the expand and collapse buttons on the left side of the metrics under the aforementioned categories to display the definition and metric-type of **Built in metric** or **Custom metric** for metrics with definitions.

Can I change the time range for metrics?

Yes for certain metrics and no for the following:

- Lead time
- Load
- Contributors
- Throughput
- Distribution

Why does the lead- and cycle-time have click in while other metrics do not?

Lead- and cycle-time are critical metrics to VSM in terms of operational effectiveness of your teams and quantifying workflow into and through the system.

How do I display charts and graphs for these metrics?

Metrics in the **Metric bar** are available in Insights on page 3170 except Bottleneck which is a measure of current process and is displayed in the **Value stream** tab. Bottleneck is explained in detailed here on page 3098.

Are custom metrics available for use?

Yes, creating custom metrics will show up on this list and be hardcoded time duration of X.

What are these custom metrics representing?

Reference previous answer and it is not directly tied to a single integration. For example, you can not view **Unit Tests** for a single integration because it applies to the whole value stream.

What does blank or zero metrics mean?

This could mean many things. There may be a dependency on lead time or the data is not linked.

How do I set up lead- and cycle-time metrics?

This is set up in the VSM JSON file on page 3059.

What metrics change when I change my definition of lead time?

Value stream metrics including the following:

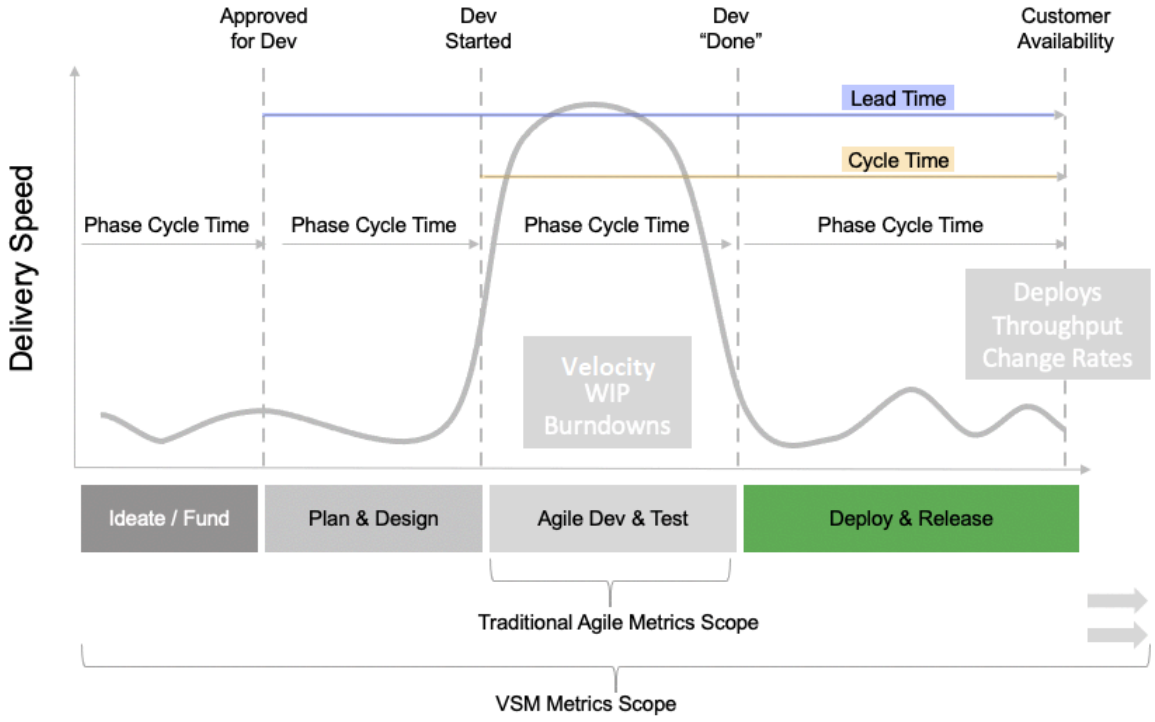
- Average Load
- Throughput
- Distribution

Can I set up multiple lead- and cycle-time metrics.

Not currently.

What is lead- and cycle-time and how are the metrics represented graphically?

Lead- and cycle-time definitions are detailed here on page 3170 and the graph shown in the following figure.



If I have a definition of lead time, can I change the algorithm?

Not currently.

Why are there similarly named metrics in the Value Streams and Insights pages?

Metric name equivalences are detailed here. on page 3084

Navigating value streams

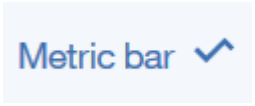
Manage a value stream's displayed data by applying filters and queries.

Value stream visualization provides an overview of the value stream workflow and summary information about the individual items contained in it. The items, sometimes called dots, represent objects from the tools that you use, such as builds, pull requests, issues, and tests. You use the To see objects from your integrated tools, apply filters by using the DevOps query language on page 3117.

You can view value stream data in a user-focused way by using the swim lanes feature. on page 3113

The highest level of a value stream is a *phase*. Phases contain the *stages* that define process flow within the phase. Stages contain data from integrated tools. Each data point is represented by an individual, graphical element. A item's position in the value stream conveys important information about the object. items in a stage named *Merged*, for example, can represent items merged into Git repositories. An item's shape and color convey information about the item's type and status. A item outlined in red, for example, might mean the item is past schedule. The border around a stage indicates how quickly stage items updated. When you hover the mouse over a item, the displayed card contains summary information about the item and links to associated tools.

In outline, to use a value stream, complete the following steps:

1. On the Measure dashboard, click **Value Streams**.
2. On the **Value streams** page, select a value stream. By default, each value stream is displayed on a separate card. You can display a list of value streams instead. The value stream is displayed on the **Value streams/value-stream-name** page.
3. Optionally, click 

to display lead time, cycle time, and devcycle time on page 3064.
4. Filter the view. The items that appear on the value stream are determined by the tools that are integrated into the value stream, and queries that filter data from the integrations. Filtering options and search scope are described later in this topic.
5. Optionally, click and select **Edit value stream map** to modify the value stream and add queries defined in the `value-stream-name.json` file.

Understanding value stream data

Queries that set the value stream scope are defined in the `value-stream-name.json` file. When you display a value stream, items matching the preset queries are displayed. The following graphic displays a value stream with items integrated from an issue tracking system and a source control system. Each stage has a query defined for it that determines the items that are displayed in it. The In Progress stage has a query defined for it similar to this one: `"query": "issue.status=\"In Progress \" and pr.status!=MERGED and pr.status!=closed"`.

You can add a global query to any value stream. The global query is applicable for all phases of the value stream. To add the global query, perform the following steps:

1. Click any value stream where you want to add the global query.
2. On the **Value stream** details page, click and then select **Update global query**.
3. Add the required DQL query to the query field of **Update Global Query** window.
4. Click **Save**.

Figure 19. User-configured value stream

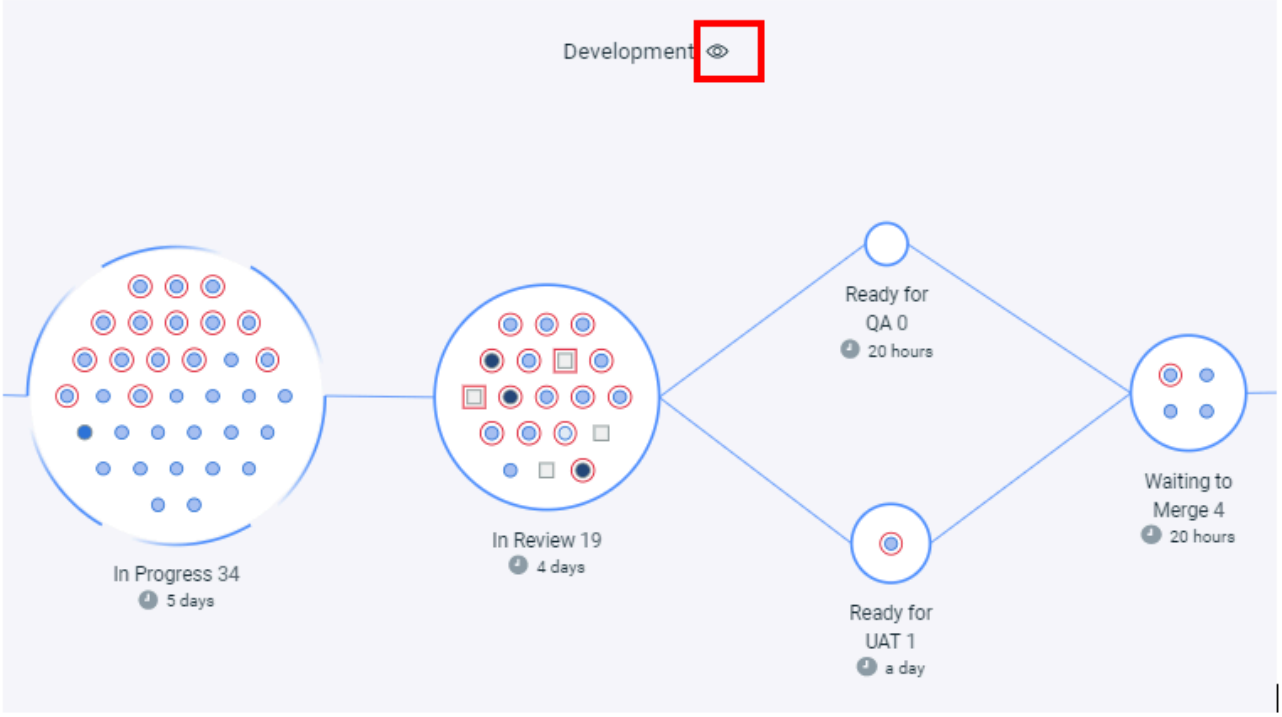
In the preceding image, the display can filtered by the following: **Time**, **Type**, **Priority**, **Release**, and **Sprint**. In addition, under each stage of the value stream, there is a value in days is called **Average wait time**. **Average wait time** is a metric that calculates the average time of dots available in any stage. The wait time metric calculates the difference between the dot transition date to any stage and the current date. Average wait time is average of the all dots wait time dvidided by number of dots. Using this metric, you can identify the dots movement through any stage and helps to take an informed decision to improve the efficiency of the project.

Note: Click **Legend** to view data on **Value stream** page for each filter selection.

If a stage has more than 40 items, summary information replaces the individual items, as shown in the following graphic:

Figure 20. Summary stage data

Use eye icon ( ) to expand or collapse the phases in the value stream.



You can define stage queries in `value-stream-name.json` file to monitor one or more branches of git repository in the value stream.

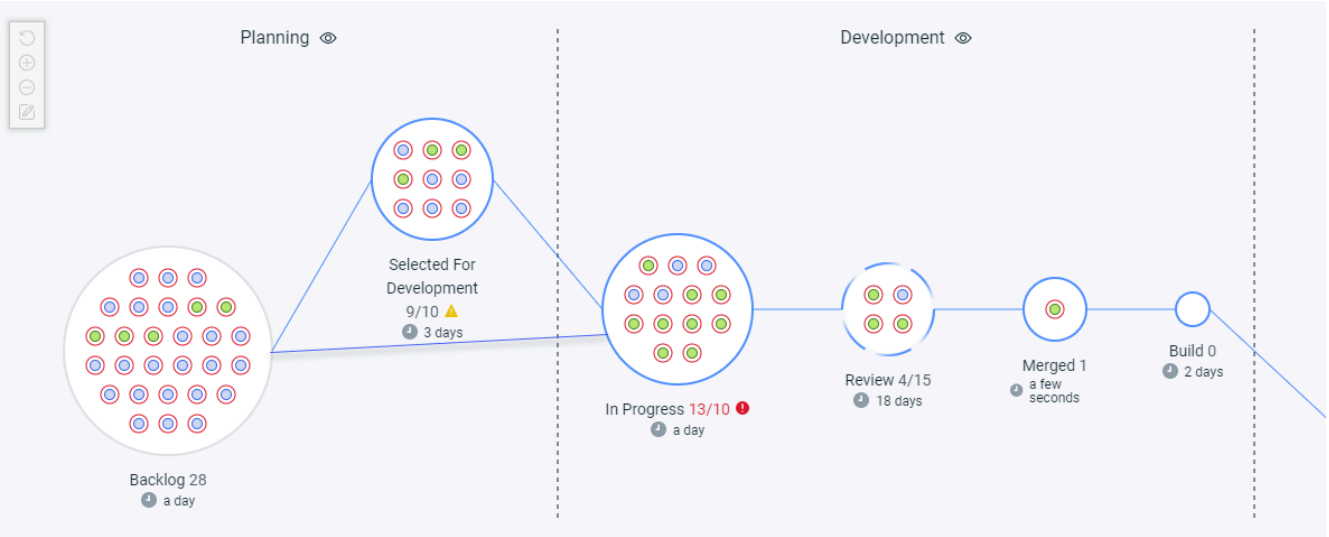
Value stream multi-branching

You can use value stream multi-branching feature to create multiple branches of the stages in the product development lifecycle. You can create multiple branches from the stages using the `targets` property in `vsm.json` file. You can also create multi-branching between the phases of the product development lifecycle. The below code is example of creating multiple branches from the `Backlog` to `Selected For Development` and `In Progress`.

```
{
          "name": "Backlog",
          "query": "",
         "targets":["Selected For Development","In Progress"],
          "description": "",
          "wipLimit": null,
          "gates": null
        },
        {
          "name": "Sprint",
          "query": "",
          "targets":[]
          "description": "",
          "wipLimit": null,
```

```
        "gates": null
      },
{
        "name": "Review",
        "query": "",
        "targets":[]
        "description": "",
        "wipLimit": null,
        "gates": null
      }
```

Note: There is no value for `targets` property in `VSM.json` file, the stage connects to the next available stage. In the above code example, `Sprint` stage connects to `Review` stage.



You can use edit mode button to do drag and drop the stages inside the phases of the development lifecycle. Click to enable/disable the edit mode. When you enable button, eye icons (and disappears. To view eye icons, disable the button. You can use zoom in and zoom out button for the detailed view of dots

clearly. You can use reset button to default view of value stream. You can use button to zoom in, zoom out, and reset controls with the capability of switching to a non-edit view mode in value stream.

Note: The user requires create pipelines permission to do drag and drop for the stages using edit mode button.

Note: You can customize the value stream layout using the value stream multi-branching. The customized layout of value stream is applicable for all users of the value stream.

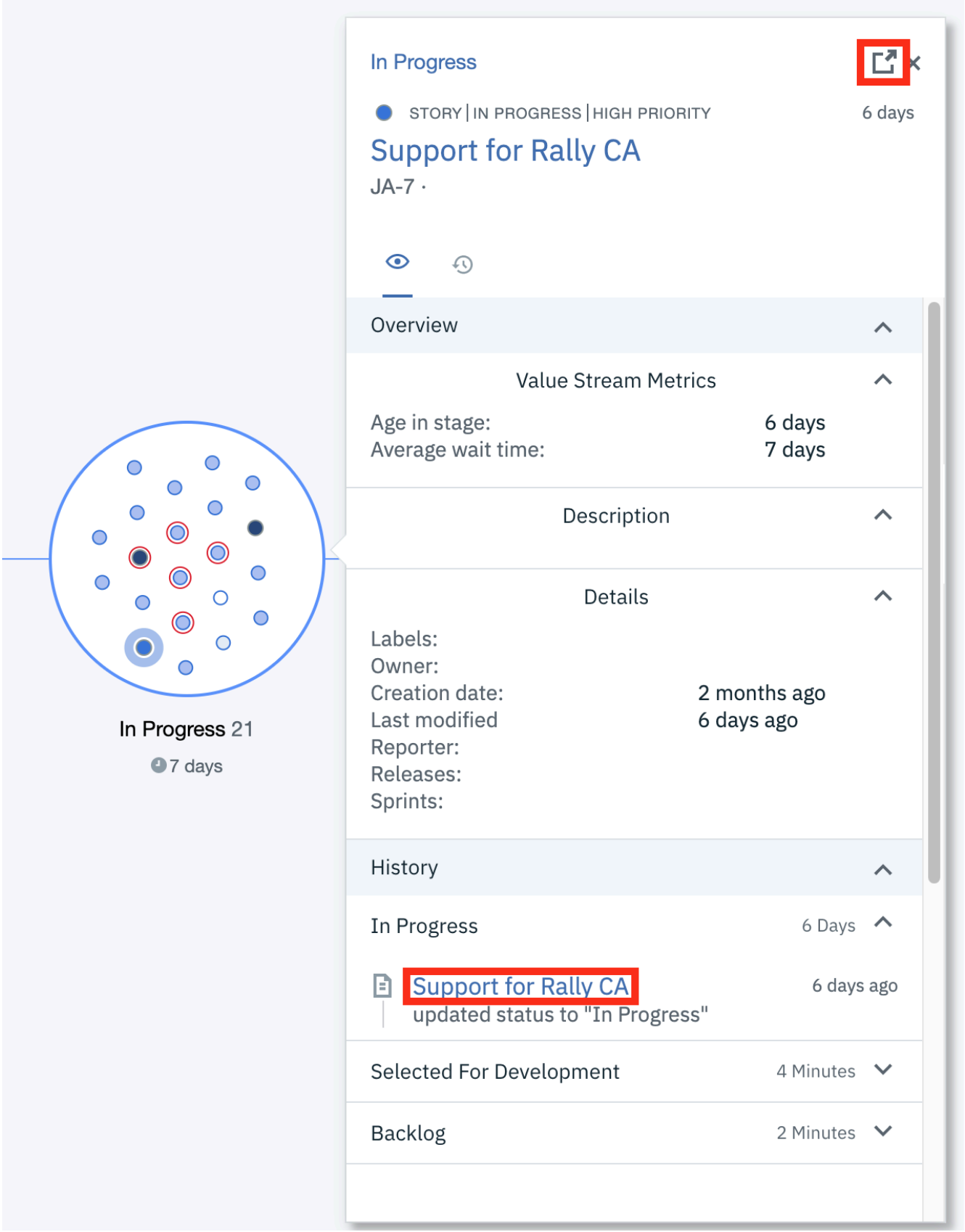
You can use the **Reset Layout** button to restore the layout of the value stream to default layout. In the default layout, all stage circles are aligned in a straight line.

Value stream items

You can display information about the items in a stage. When you click an item, a card is displayed with information about the represented item, such as a Jira issue. Full history about the item is presented in context, including its history. You can use the supplied links to open the item in originating tool.

The following graphic displays a Jira issue in the value stream's In Progress stage. Links, highlighted in red in the graphic, can open the item in the integrated tool.

Figure 21. Item card



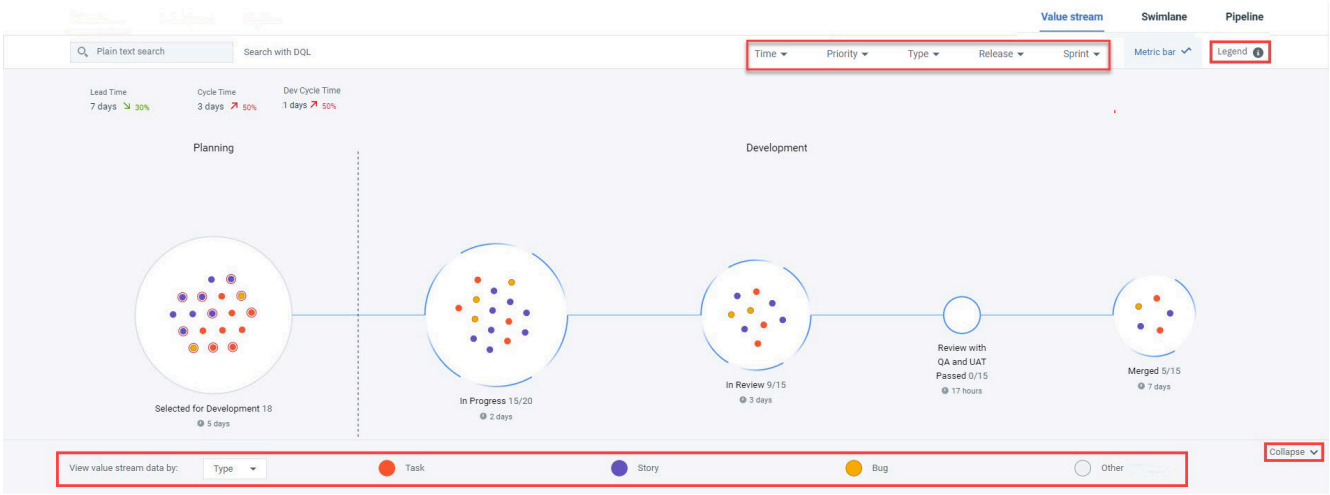
If linking rules are defined for the integrations, information from the linked tools and available on page 3066. Link rules link two integrated tools, such as Git and Jira, to make information from both tools available in the affected items.

Value stream view by

At the bottom of the **Value stream** or **Swim Lanes** pages next to the **View value stream data by** list, you can view your data by selecting one of four criteria applied to the value stream data with an associated color legend and may also be shape coded by the information detailed in the following table and figure:

Table 1162. View value stream data by

View value stream data by	Description
Priority	Level of importance such as lowest, low, medium, high, and highest.
Release	The release versions.
Sprint	Sprint numbers, names, or both.
Type	Dots type such as story, bug, task, epic, pull request, and commit
Time	The waiting time of dots in any particular stage.



Note: If the **View value stream data by** bar is not visible on either the **Value stream** or **Swim Lanes** pages, click **Expand** located toward the bottom right corner of the respective pages to display the bar. Additionally, if the bar is visible and you want to hide it, click **Collapse**.

Toward the top right corner of either the **Value stream** or **Swim Lanes** pages, click **Legend** to display additional identification details for the items including any of the aforementioned shape coded items. To view value stream data by a range of data, click the **View value stream data by** list and select your required criteria

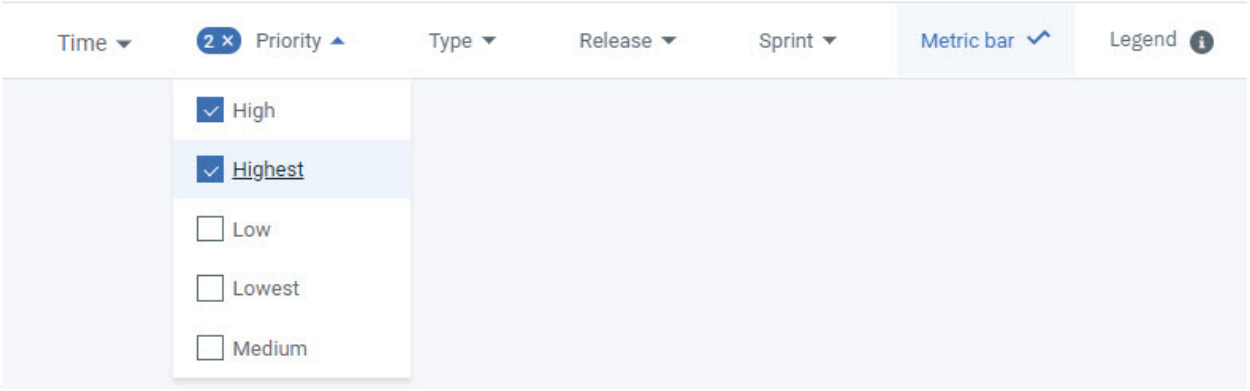
Quick filters for value stream data

At the top of the **Value stream** or **Swim Lanes** pages to the left of the **Metric bar** button shown above, you can use quick filters to limit the amount of value stream data that is displayed based on the selected criteria of priority, release, sprint, or type that is detailed in the above table.

Ensure the **View value stream data by** bar is visible on the bottom of either the **Value stream** or **Swim Lanes** pages and perform the following procedure to search and limit the display of value stream data based on your required criteria.

1. Click the **View value stream data by** list and select your required criteria. The color coded items associated with the selected criteria are displayed in the legend on the bottom bar.
2. To further limit the display of value stream data, you can apply quick filters by clicking on **Priority**, **Release**, **Sprint**, and **Type** and selecting check boxes within each item in any multiple combinations of all filters to the previously selected criteria in the bottom bar for precise narrowing of your required data range. To display the value stream data of all current sprints, select **Current Sprint(s)** check box from the **Sprint** quick filter list.

   An example of the **Priority** quick filter applied with the **High** and **Highest** criteria selected is illustrated below.



3. To remove the quick filters, either click the **x** to the left of any filters that were selected as shown above or individually deselect individual items within each of the criteria.

Dot alerts

You can see the alert for dots in value stream for the following scenarios:

1. Dots that are unassigned.
2. Dots with long wait time.

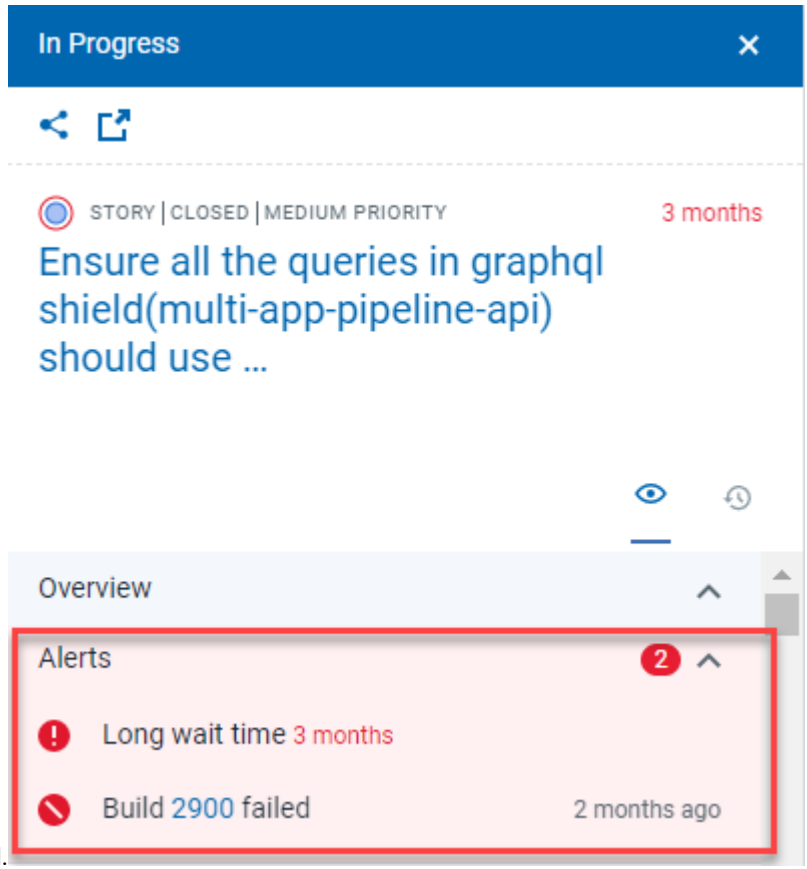
   Note: Wait time is defined as the time between the current time and the last status update of the dot. This alert is generated when a dot's wait time is greater than the average wait time across all dots. Dots alert specifies the value of waiting time.

3. Dots with build status failed.

Note: Dots alert specifies reference of the failed build.

4. Dots with deployment status failed.

The figure below is an example of dot alerts for long wait time and build failed.

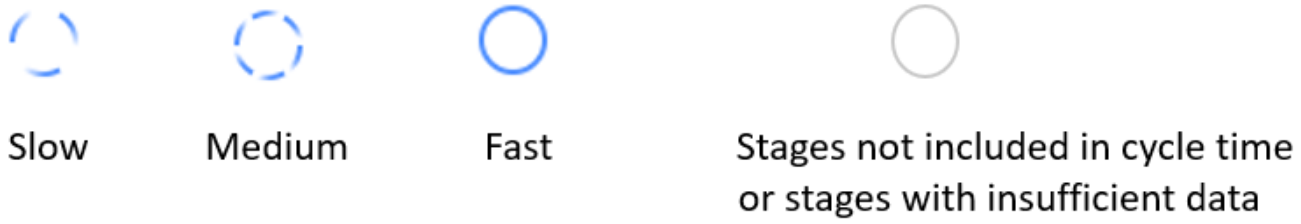


Stage wait time

Stage wait time is the average time of all the dots per workflow. There are four types of stage wait time.

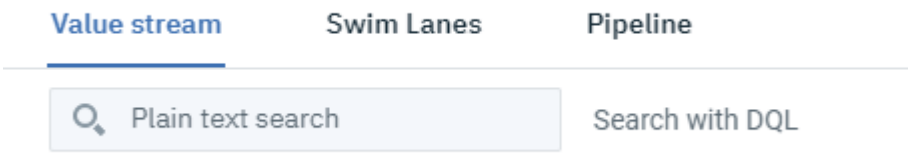
1. Blue circle with three segments - Stages with slow wait time. The dots in the stages are progressing slowly compare to other stages of the workflow.
2. Blue circle with six segments - Stages with medium wait time. The dots in the stages are progressing in medium speed when comparing with fast and slow stages.
3. Blue circle - Stages with fast wait time. The dots in the stages are progressing fast compared to other stages in the workflow.
4. Grey Circle - Stages with insufficient data.

The stage time is displayed below each stage circle.



Searching the value stream

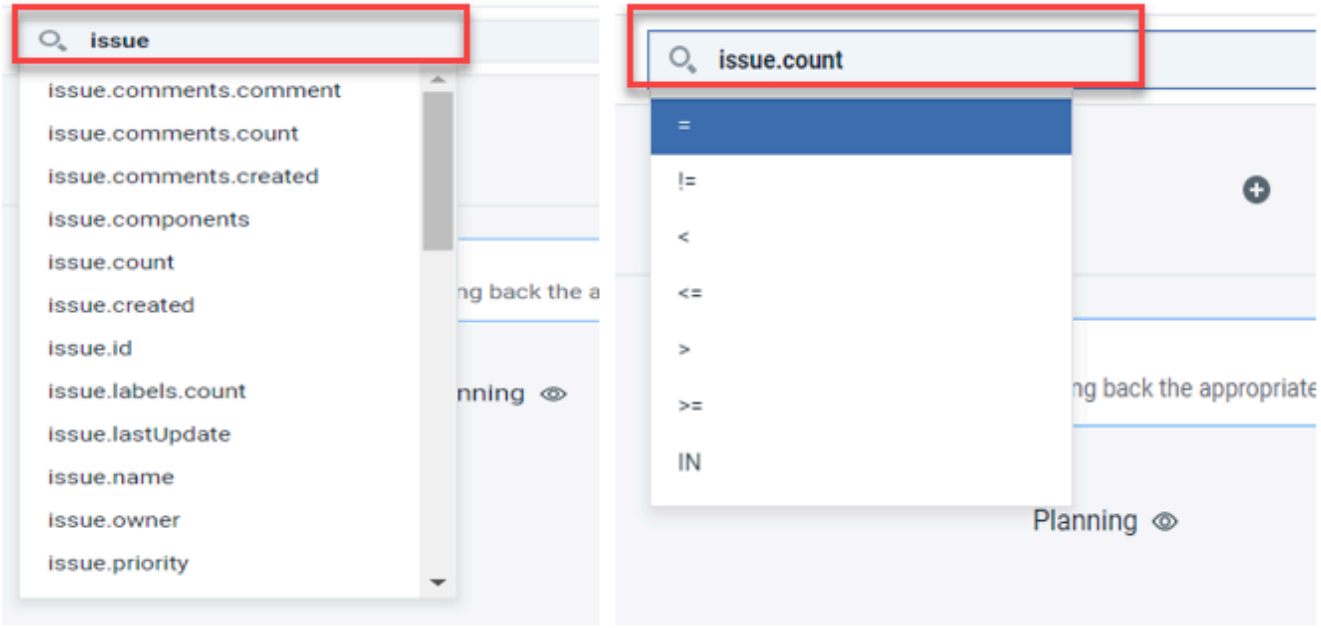
Searching a dot in value stream data is possible in two ways, **Plain text search** and **Search with**



DQL .

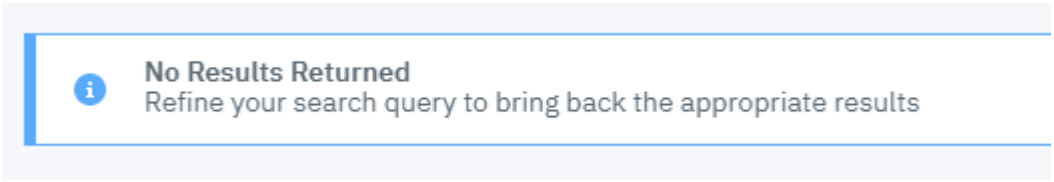
Plain text search is the free text search of the dot in the value stream. When search text is entered in **Plain text search** field, it displays all dots which are associated or contains the search text. Use **Plain text search** field to search the dots with issue id, issue labels, issue name, issue owner, issue priority, issue types, releases, and sprints.

Use **Search with DQL** field to search the dots with DQL queries on page 3118 and DQL operators on page 3121. The **Search with DQL** field displays all possible queries and operators for the typed



query.

Note: In case, no dots is available for the search data, value stream will display a error message.



Managing swim lanes

Use swim lanes to display a user-centered view of your value stream data.

About this task

You can use the dynamic swim lanes feature to quickly display vital project information across various organizational roles enabling informed workflow decisions. Dynamic swim lanes are adaptable and provide tailored data sets according to the different roles and requirements detailed in the following table.

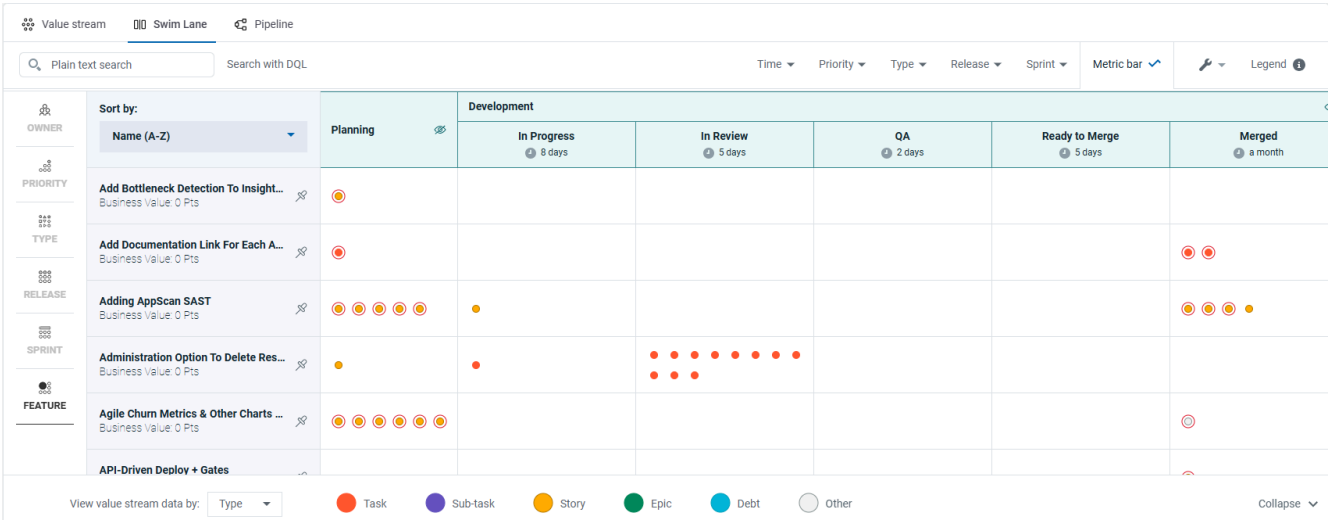
Role	Dynamic swim lanes view
Developers, scrum masters, daily stand-up	View work items per person, show and hide selected stages, and choose sprint view for displaying pointed data sets.
Development managers, director	Check overall sprint status, view stage position of work items, display remaining work from past sprints for possible future sprint allocation, and click **SPRINT** for a consolidated work item view displaying the current status.
Scrum masters, development managers	For tracking targeted Epics, use the pinning feature to keep the prioritized items at the top of the list.
All roles	Dynamic swim lanes display current state of releases, contents of both upcoming and past releases, determine when work items are released, view work items per stage, and click **RELEASE** for a detailed release contents list and click **SPRINT** for the same result.

A swim lane organizes the value stream into a table. The table columns represent value stream phases and associated stages shown above the table data in the illustration below, and the rows represent either:

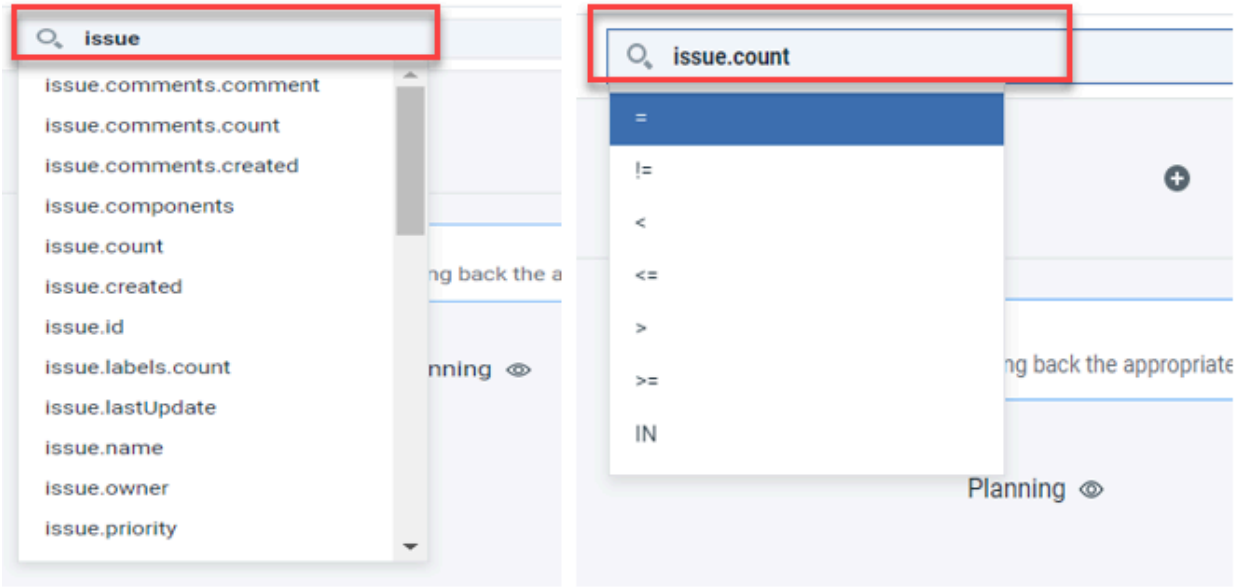
- Owner (all rows displayed)
- Priority (all rows displayed)
- Type (all rows displayed)
- Release (five rows displayed)
- Sprint (five rows displayed)
- Feature(all rows displayed)

properties, as shown in the dynamic row selector on the left side of the following illustration:

Figure 22. Dynamic swim lanes



Note: Based on the property selected in the dynamic row selector, you can search for required information based on the selected property type in the search field located toward the top right of the dynamic row selector shown in the illustration above. Additionally, you can perform DQL queries attached to the value stream on page 3117 by clicking **Search with DQL** field where you can enter the DQL query. The **Search with DQL** field displays all possible queries and operators for the typed

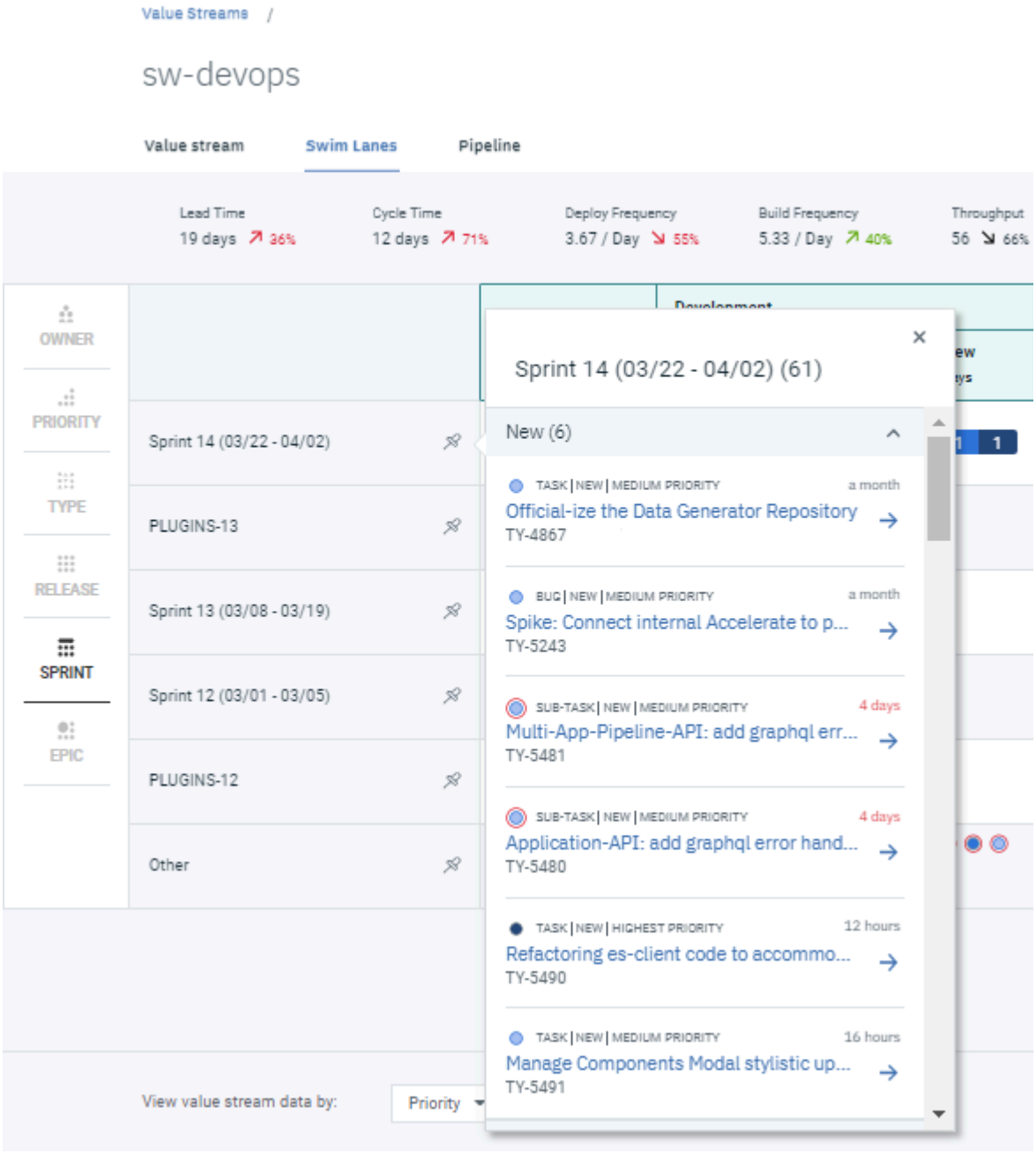


query.

Dynamic swimlanes works with filters and legends detailed on this page on page 3103. If a row does not have dots, it will not display when you filter.

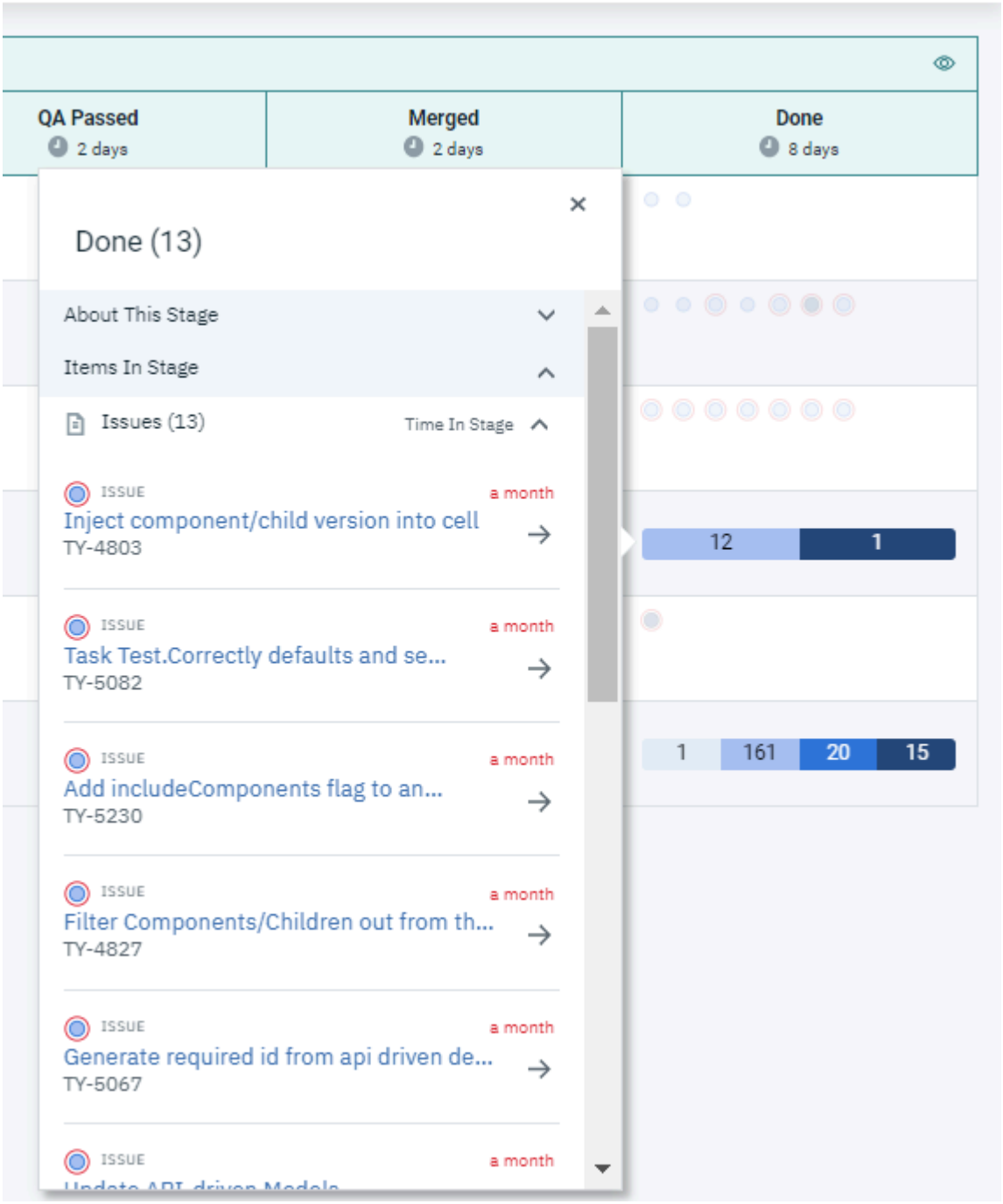
On the right side of the dynamic row selector is the rows overview panel. The row contents vary according to the specific property selected in the dynamic row selector and you can click on any row to display the row panel shown on the right of the

rows overview panel where you can view dots in each stage for a specific row. You can also expand and collapse sections in the row panel by clicking the down and up arrows on the right of the section names as shown in the following illustration.



You can show and hide phases and associated stages by clicking and respectively with both located on the right side of the phase names in the column fields above the data table. When you hide phases, a high-level progression view is displayed. When you show phases, a detailed view of items in stages is displayed. Also, when you click on the phase names in the column fields above the data table, a row panel is displayed containing the associated stages with the phase. You can also expand and collapse stage sections to view phase dots according to the stage position in the phase.

Additionally, you can click on any of the summary bars in the data table to display a row panel with the same aformentioned features as shown in the following illustration.



You can prioritize items for all row categories in the row overview panel by clicking for the required row and it remains at the top of the row overview panel with last item pinned on top. To unpin an item, click to remove the priority item.

Note: The pinning preferences, phase collapse or expand, drop-down filters, and row views are all saved to the user settings for VSM.

The functionally available on the **Value stream** view is available here as well; you can access the **Metric bar** and text and DQL query field.

To display a swimlane, complete the following steps:

1. On the Measures **Home** page, click **Value Streams**, and then select a value stream.
2. On the value stream page, click **Swim Lane**.
   Result
   The swimlane is displayed.

DevOps query language (DQL)

Use the DevOps query language to filter the information displayed in your value stream.

About the DevOps query language

The DevOps query language is a search tool designed to filter value streams. You build simple queries by combining searchable items with operators and values. A searchable item is a query-able object, such as `issue.status`. Nearly all Measure database fields are query-able, and you can search across nested objects, for example, `issue.project.name`.

Query scope

Queries can be scoped to the entire value stream or a stage. You can create queries on the **Value stream** view or define them in the `value_stream_name-vsm.json` file attached to a value stream. Each stage can have a query defined for it.

You set the scope of a query at the value stream level by defining a query in the header of the `value_stream_name-vsm.json` file or on the **Value stream** view. You might do this if you have a shared Jira® instance, for example, and want to restrict the displayed issues to those that belong to your team. For example, `issue.project.name="my_jira_project"`.

If you set the scope to the stream level, other queries work within that scope. After you set the scope to the value stream level, you cannot override it by defining a new stream-level query on the **Value stream** view. To change a value stream scoped query, replace the query in the `value_stream_name-vsm.json` file.

Simple queries

Filter your value streams with queries that restrict the displayed information to the things that you want to see. You build simple queries with the DevOps query language by combining searchable items with operators and values. A searchable item is a query-able object, such as `issue.status`. The query format consists of an item, a period, a query-able parameter, an operator, and a value of a valid type:

```
[item].[field] [operator] [value type]
```

The following list displays some typical queries:

```
issue.status = “In Progress”
issue.owner = “John Doe”
pr.number > 5
issue.status != Done
build.endTime < 1d
issue.created > 10d
issues.sprints.name = "sprint 1"
```

Value types are case insensitive. Text value types can be enclosed in single- or double-quotes.

A list of searchable objects, or items, are provided later in this topic.

Combining queries

You can combine simple queries by using the logical operators ‘and’ and ‘or’. Combined queries are resolved in left-right order.

The following query displays issues for Jane and John Doe with the status of 'In Progress' .

```
issue.owner = “John Doe” or issue.owner = “Jane Doe” and issue.status=“In Progress”
```

The following query displays items with the status of "Open," and that are older than one day.

```
pr.status = Open and pr.created > 1d
```

Priority queries

You can override the default order of precedence by using parenthesis.

Unlike the earlier example, this example displays all issues owned by John Doe, and only issues owned by Jane Doe with the status of "Done."

```
issue.owner = “John Doe” or (issue.owner = “Jane Doe” and issue.status=Done)
```

This example displays John Doe's open issues and with the status of "In Progress."

```
(issue.status = “In Progress” or pr.status = open) and issue.owner = “John Doe”
```

Elements of the DevOps query language

Use items in table to search in their query-able fields. Time-type fields are shown in *italicized* font.

The below table lists all the DevOps queries that you can use in stages of `value_stream_name-vsm.json` file. Using the query, you can view the data of an application integrated in value stream.

Table 1. Searchable items - stages of `value_stream_name-vsm.json`

Table 1163. Searchable items - stages of `value_stream_name-vsm.json`

Item	Query-able fields
issue	`count`, `status`, `labels`
issue.comments	`count`

Table 1163. Searchable items - stages of `value_stream_name-vsm.json` (continued)

Item	Query-able fields
issue.extended	<custom field>
issue.labels	`count`
issue.project	`id`, `key`, `name`. For example, `issue.project.name="my_project"`. Note: A project is a group of issues that include Jira: stories, bugs, tasks, and epics for a product, service, or activity.
issue.sprints	`active`, `count`
pr	`count`, `draft`, `labels`, `repositoryId`, `repositoryName`, `repositoryOwner`, `source`, `status`. For example, `pr.draft=true` will show only the issues that is having the draft pull requests in the value stream.
pr.approvals	`count`
pr.assignees	`count`
pr.comments	`count`
pr.labels	`count`
pr.reviewers	`count`
pr.reviews	`count`
commit	`count`, `repositoryId`, `repositoryName`, `repositoryOwner`, `source`
build	`count`, `status`, `type`
deployment	`count`, `env`, `environmentId`, `source`, `status`

The below table lists all the DevOps queries that you can use to search in **Value stream** view. Using the query, you can search in **search bar** to filter the required dots in **Value stream** view.

*Table 2. Searchable items - **Value stream** view search bar*

Table 1164. Searchable items - Value stream view search bar

Item	Query-able fields
build	`count`, `id`, `name`, `source`, `status`, `endTime`, `requestor`, `startTime`, `number`, `type`
build.application	`id`
commit	`name`, `repositoryId`, `repositoryName`, `repositoryOwner`, `source`

Table 1164. Searchable items - Value stream view search bar (continued)

Item	Query-able fields
deployment	`count`, `id`, `name`, `source`, `status`, `endTime`, `requestor`, `startTime`, `number`, `env`, `environmentId`, `versionName`
feature	`created`, `creator`, `id`, `name`, `status`, `url`
issue	`count`, `components`, `id`, `impact`, `name`, `storyPoints`, `source`, `status`, `owner`, `priority`, `type`, `created`, `lastUpdate`, `resolution`, `resolved`, `severity`, `labels`, `businessValue`
issue.comments	`count`, `comment`, `created`
issue.extended	<custom field>
issue.labels	`count`
issue.project	`id`, `key`, `name`. For example, `issue.project.name="my_project"`. Note: A project is a group of issues that include Jira: stories, bugs, tasks, and epics for a product, service, or activity.
issue.rawIssue	The `rawIssue` field contains non-normalized data from the external tool, such as Jira. For example, `issue.rawIssue.fields.custom_field`.
issue.releases	`id`, `name`, `description`, `archived`, `released`. Several typical examples include, `issue.releases.name="Version 1.1"`, `issue.releases.released = false` . Note: A release is a deployment to an environment.
issue.sprints	`active`, `count`, `id`, `name`, `previous`, `description`, `startTime`, `endTime`, `completeTime`, `url`
issues.comments	`updated`
issues.releases	`releaseDate`
pr	`count`, `draft`, `id`, `issueIds`, `labels`, `name`, `source`, `status`, `branch`, `baseBranch`, `commitId`, `merged`, `assignees`, `reviewers`, `created`, `lastUpdate`, `creator`, `number`, `repositoryId`, `repositoryName`. For example, `pr.draft=true` will filter only the issues that is having the draft pull requests in the value stream.
pr.approvals	`count`
pr.assignees	`count`

Table 1164. Searchable items - Value stream view search bar (continued)

Item	Query-able fields
pr.comments	`count`, `comment`, `created`, `updated`
pr.labels	`count`
pr.reviewers	`count`
pr.reviews	`count`, `status`

Note: A feature is a collection of stories and equivalent to an Epic in Jira.

The query operator tables displays search operators.

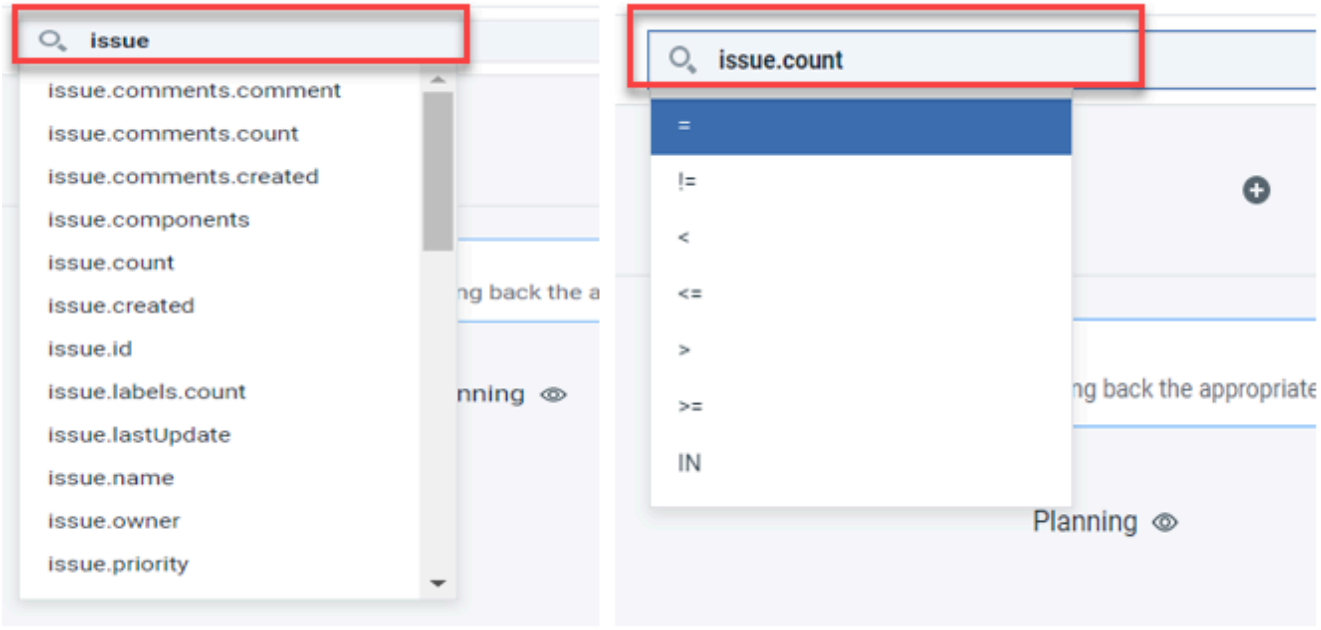
Table 1165. Query operators

Operator	Definition
=	equal
>	greater then
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal
""	required for exact phrase match with spaces
( )	order of operations grouping
`AND` or `and`	conjunction for searching both values
`OR` or `or`	disjunction for searching one value or another
`IN`	disjunction for searching one value or another.

Table 1165. Query operators (continued)

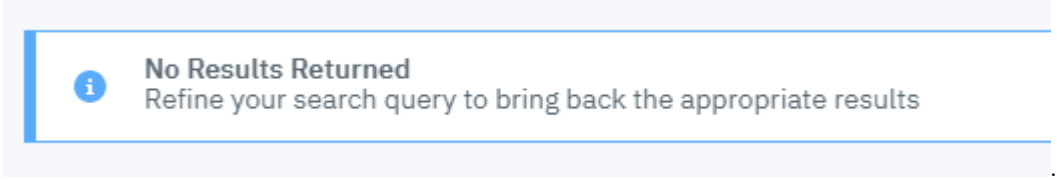
Operator	Definition
	Note: The `IN` operator is shorthand for the `OR` operator. Use the `IN` operator to avoid using multiple `OR` operators when you query in the same field.

Use **Search with DQL** field to search the dots with DQL queries on page 3118 and DQL operators on page 3121. The **Search with DQL** field displays all possible queries and operators for the typed



query.

Note: In case, no dots is available for the search data, value stream will display a error message.



.

The value types table describes the valid value types.

Table 1166. Value types

Value type	Definition
value	Supports any characters except whitespace or the following reserved characters : . = ! < > " ( )

Table 1166. Value types (continued)

Value type	Definition
quoted value	Characters are wrapped in double quotes. This type supports any character including those unsupported by value. For example: `“This is a quoted value”, “(This = valid input!)”`
time	Use to compare time fields. Valid input is a number followed by a measure of time, `(h = hours, d = days, w = weeks, y = years`. For example:`1w (one week), 10d (ten days), 3y (three years).`

External YAML configuration

You can create a minimal value realization of different DevOps tools into Measure quickly using external YAML configuration file.

You must define all your required configuration properties in `ucv.yaml` file, then you can import the `ucv.yaml` into Measure and apply the configurations defined in the YAML file.

You must perform the following steps to see the minimal value realization of DevOps tools into Measure:

Creating the YAML configuration file in GitHub repository

You can create YAML configuration with the properties of required DevOps tools.

The following table describes the properties required to define the configuration in the YAML file.

Property name	Type	Description	Required
version	String	The version number of the YAML file.	Yes
integrations	Array	List of integration configurations	Yes

Integrations

You must define the integration properties that include **type**, **attributes**, and **configs.** The following table describe the details of **type**, **attributes**, and **configs** properties:

Property name	Type	Description	Required
type	String	Type or category of the CI/CD tool.	Yes

Property name	Type	Description	Required
attributes	Object	Additional attributes specific to the integration.	Yes
configs	Object	Configuration object for the integration.	Yes

Configs

The following table describes the properties that you must define under the configs property:

Property name	Type	Description	Required
name	String	Name of the configuration.	Yes
loggingLevel	String	Logging level for the integration.	Yes
deltaTime	String	Time difference for the integration.	Yes

Depending on the integration type, additional configuration properties might be required. For more information about the additional configuration properties, see Measure plug-in configuration.

Sample YAML configuration file

The following sample shows a YAML configuration file with two integrations:

```
version: "8.0.0"
integrations:
  - type: ucv-ext-deploy
    attributes:
      auto-config: true
    configs:
      name: Deploy Integration
      loggingLevel: DEBUG
      deltaTime: '2023-05-15T18:30:00.000Z'
      # Additional integration-specific fields for Deploy Integration

  - type: another-integration
    attributes:
      auto-config: true
    configs:
      name: Another Integration
      loggingLevel: INFO
      # Additional integration-specific fields for Another Integration
```

Importing YAML configuration file to Measure

The YAML configuration file must be imported into Measure before you can apply the configurations defined in the YAML file.

You can import the YAML configuration file into Measure by any one of the following methods:

Importing YAML configuration file using GitHub plug-in integration

You can import the YAML configuration file into HCL DevOps Velocity (Velocity) by integrating GitHub repository that contains the ucv.yaml file.

1. Log in to Velocity.
2. Click **Settings > Integrations**.
3. In the Installed tab, click **Add Integration** for the installed GitHub plug-in.

   Note:
   - If the GitHub plug-in is not installed, go to Available tab and click Install for the GitHub plug-in.
   - Ensure that you add integration with GitHub plugin version 1.4.5 or later to sync the YAML configuration file.

4. Enter the details in the required fields to add the GitHub integration.
5. Click **Add** to save the details and enable the integration.

   Result

   After integration, the YAML configuration file from the GitHub repository is automatically synced to Velocity and displayed under **Settings > External config pipeline**.

   Note: If you want to make changes to YAML configuration file, you can update the file in GitHub repository and it is automatically synced to Velocity.

Results

You have successfully imported the YAML configuration file from the GitHub repository into Velocity.

Importing YAML configuration file using user interface

You can import the YAML configuration file into HCL DevOps Velocity (Velocity) using user interface if it is not hosted in a GitHub repository.

1. Log in to Velocity.
2. Click **Settings > External Config Pipeline**.
3. Click **Download sample yaml** to download the sample YAML file.

   Result

   The sampleYamlFile.yaml file is downloaded.
4. Edit the sample file with the actual values of your configuration.
5. Click **Add new config file**.

   Result

   Add New Config dialog opens.
6. Click **Choose file** to select the modified file.

7. Click **Confirm** to upload the file.
   Result
   The modified YAML configuration file is displayed in the configuration files list.
8. To modify the existing YAML configuration file, perform the following steps:

   a. Click the overflow menu for the required YAML configuration file and select **Edit**.

   Note: You cannot make changes to YAML configuration file that is imported through GitHub integration from UI, to make changes you can update the file in GitHub repository and it is automatically synced toVelocity.

   Result
   The existing YAML configuration file is downloaded.

   b. Edit the downloaded YAML configuration file.

   c. Click the overflow menu for the YAML configuration file which you have edited and then click **Update**.
   Result
   Update config dialog opens.

   d. Click **Choose file** to select the edited file.

   e. Click **Confirm** to upload the file.
   Result
   Yaml config added successfully message is displayed.

Results

You have successfully imported the YAML configuration file into Velocity form user interface.

Applying the YAML configuration file to Measure

You must apply the YAML configuration file to Measure for viewing the configuration defined in YAML file.

About this task

After you import the YAML configuration file, you can apply the configuration to Measure.

1. Log in to Measure.
2. Click **Settings > External Config file**.
3. Choose the required YAML configuration file in the list and click the right angle bracket to preview the configuration details.

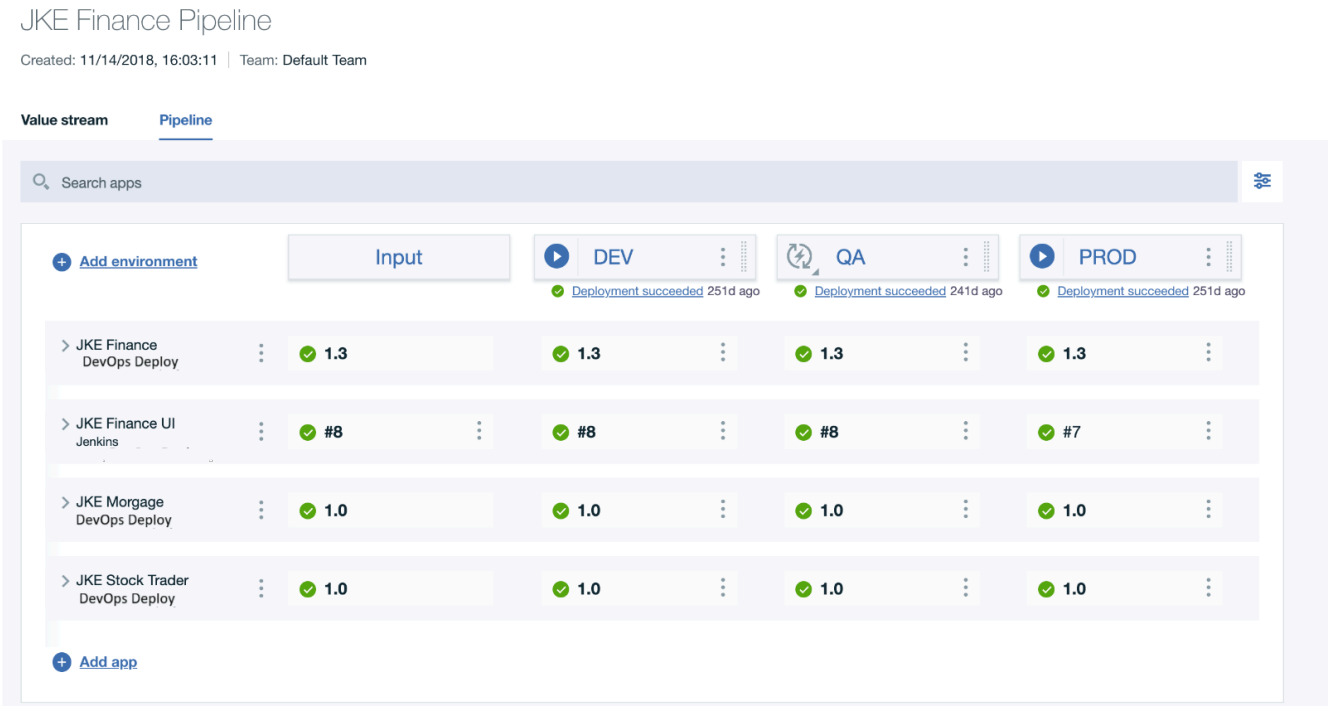
4. Click **Apply** to apply the required YAML configuration file to Velocity.
5. Click the overflow menu ⋮ for the required YAML configuration file and then click **View logs** to view logs for the integration.

Pipelines

Pipelines provide a streamlined method to manage the applications in your development lifecycle. With pipelines, you can auto-generate deployment plans and deploy applications versions to all environments.

A pipeline represents the environments and applications that you use to drive your development lifecycle. Environments are logical groupings for applications and tools. This process is intended to increase application quality before delivery to the target environment. A typical pipeline is shown in the following graphic:

Figure 23. Pipeline



Pipeline features include the following items:

- A pipeline represents the applications integrated into the value stream from Measure or Jenkins.
- The top row identifies the environments and provides tools to manage the environment.
- Each column represents a logical environment.
- The rows represent application versions. You can drill down to the underlying applications to view releases, history, or to troubleshoot.
- You can use the play button at the top of an environment to manually start a deployment. By default, this action deploys all changed applications in parallel.

- You can schedule deployments.
- You can add gates to environments that require approval before application versions can run in the environment.

Note: Ongoing active pipeline improvements are in process for robust performance capabilities including the following:

- Running deployments through the pipeline is documented here on page 3153 and there are gaps between the currently documented features and capabilities of these new plugins.
- Externalizing new deployment CI/CD plugins integrating with Measure.
- Performing deployments using the Deploy and Azure DevOps plugins and greater flexibility.

Pipelines accept input from either source control management (SCM) repositories or other external applications, such as Deploy. When you create an environment, some default settings are set for you on the **Input** environment.

You add applications to the pipeline from external tools that are integrated with the product, such as Jenkins™. Jobs run serially; they enable flow control for your work. For example, you might run a job in a test environment before deploying to the production environment. You can ensure that if the tests fail, the deployment job won't run.

You can add gates to environments. A gate is a condition that must be met before jobs can run in the environment. For example, you might add a gate that requires an approval before a job can start.

You can define environment properties that can be used in all jobs. For example, you might define a `TEST_URL` property that passes a single URL to deploy and test jobs in a single environment. The deploy job would deploy to that URL, and the test job would test the running app at the URL.

Each pipeline environment automatically has a release and deployment plan associated with it. You do not have to manually create a release beforehand. As you add applications to an environment, a task is automatically added to the default deployment plan.

When you run a pipeline application, a deployment starts that uses the associated deployment plan. You can monitor the progress of the deployment with links provided on the stage card.

Next to each pipeline, you can see a Value Stream icon. Click the icon to view the corresponding Value Stream page.

Managing pipelines

Add applications and jobs to your pipeline, and add or modify stages.

About this task

A pipeline is created when you create a value stream. If, for example, you create a value stream named "My Value Stream," a corresponding "My Value Stream" pipeline is also created. On the Home page, click Pipeline icon to go to the pipeline page.

You can add environments at any time and rearrange them as needed. Each application that you add to a pipeline is inserted onto a separate row below the last application. As you add apps and jobs to stages, tasks are added to the associated deployment plan in the order that they appear in the pipeline.

To modify a pipeline, complete the following steps:

1. **Optional:** On the **Pipeline** page, click **Add Environment** to create an environment, or stage.

    a. In the **Add environment** window, and in the **Environment name** field, enter the environment name.

    b. In the **Action** field, select the check boxes for the required deployment mode from the list of **Deploy**, **Join release**, and **Schedule deployment**.
    If more than one deployment mode is selected, both the default deployment mode play control and drop-down will be visible on the environment.

    The deployment mode play controls are as follows:

    - **Deploy**
    - **Join release**
    - **Schedule deployment**

    c. Based on the selected mode, in the **SET AS DEFAULT** field, select the default deployment mode.
    The deployment mode play control that is visible on the environment is the default.

    d. Click **Save**.

    e. Move the environment to the required location in the pipeline by hovering the mouse over the right-side of the environment, and then drag it to the required location in the pipeline.
    You can re-order the environments except for the first stage, **Input**.

    f. Click and select **Edit environment** to update the following parameters:

Parameter	Description
Environment name	The name of the environment.
Permitted actions	The actions to perform in environment. Possible actions are **Deploy**, **Join release**, and **Schedule deployment**.You can set any of these actions as **Default action**.
Groups and users allowed to perform actions	The group or user to perform the deployment in environment.

Parameter	Description
	Note: Only the selected group or user can perform the deployment in the environment.

In the **Edit environment** screen, select the groups or users who have permission to deploy in the selected environment.

Note: The group or user only can deploy to the selected environment.

g. Click ⋮ and select **Override inventory** to override the application in the deployment environment of pipeline. In the **Override inventory** screen, select the version for the application to override deployment in the selected environment. You can use **Override inventory** in all environments except **Input**.

2. **Optional:** To map an application environment to the pipeline, complete the following steps:

   a. For the environment where you want to add the application, click + **Add app**.

   b. On the **Add apps - Choose the apps** window, in the **Managed by** list, select the required application.

   c. Click **Save**.
   Result
   The application is added to the stage.

   d. To view the deployment template associated with the stage, click ⋮ and select **View deployment template**.
   Result
   The deployment plan detail page is displayed. As you add apps and jobs to the pipeline, associated tasks are inserted into the plan. You can also add tasks directly to the plan.

   e. To modify the mapped environment, click ⋮ and select **Edit application environments**.

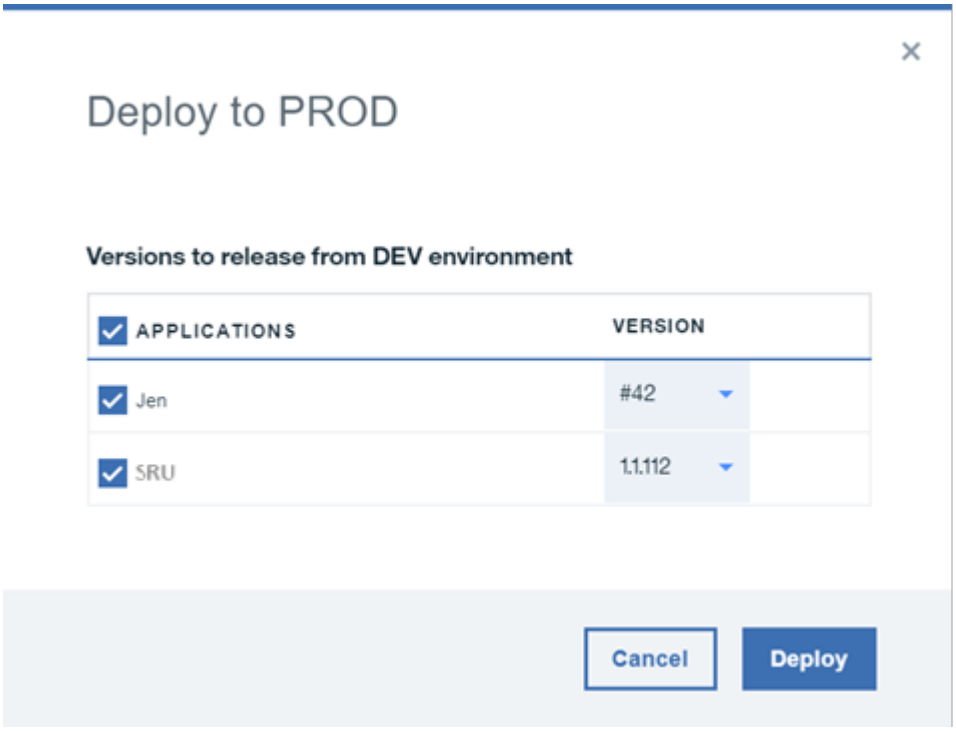
   f. To delete the environment, click ⋮ and select **Delete**.

3. **Optional:** To edit HCL DevOps Deploy (Deploy) application components already created on the pipeline, click ⋮ on the right side of the application and select **Manage components** to add and remove components of the application, **Edit description** to modify the description of the application in the selected pipeline, or **Delete application** to delete the application in the selected pipeline.
4. **Optional:** To add a gate on page 3145, click **Add gate** and then define the gate rule.
5. **Optional:** Click on the deployment mode play control to open the respective modal window.

The applications along with the versions deployed or joined releases in the previous environment will be available in the modal window dialogs shown in the figures below. The version to be deployed or joined releases for the selected application can be selected from the drop-down lists shown in the figures below.

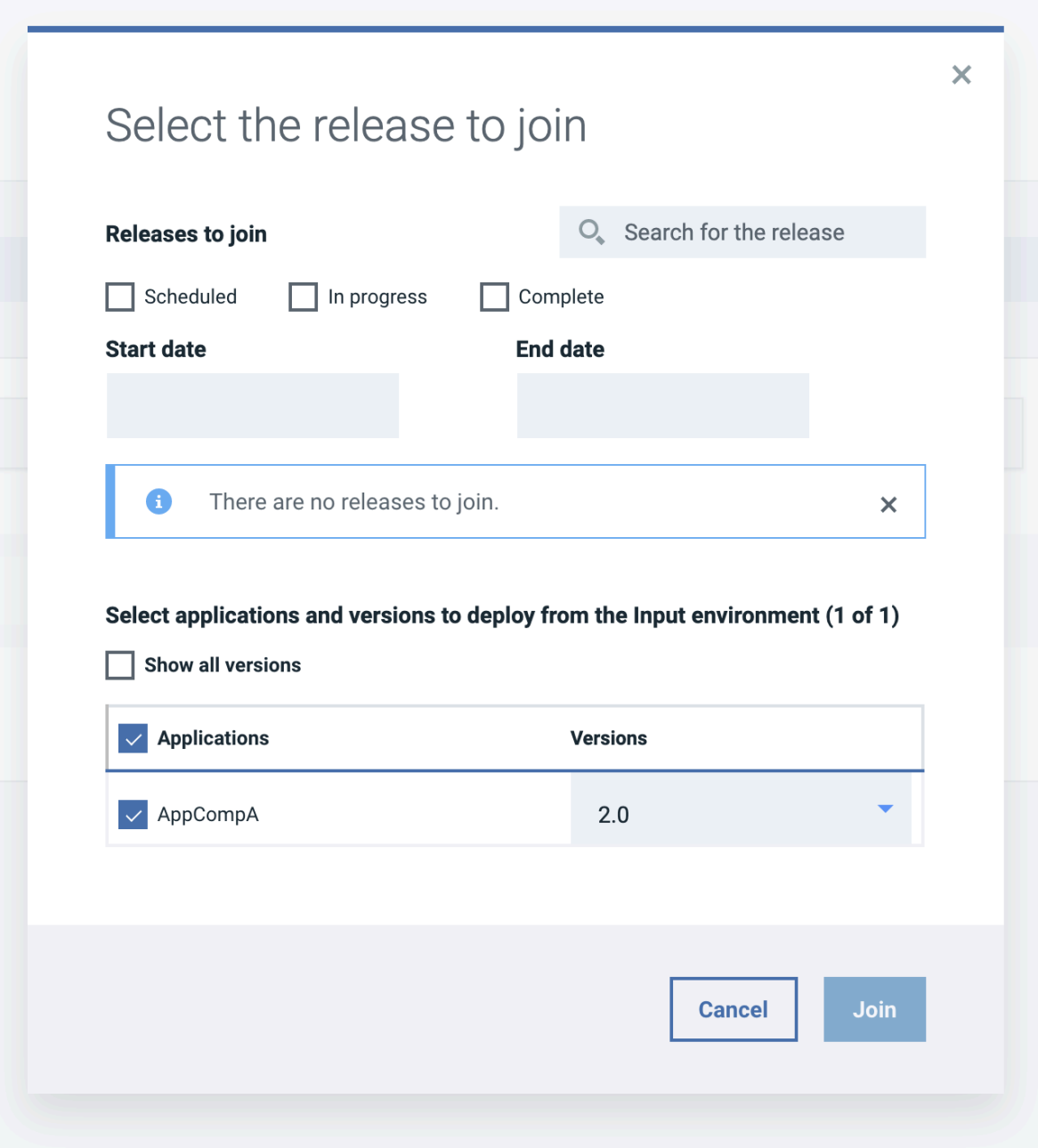
a. If **Deploy** is selected as the default deployment mode, the **Deploy** modal window is displayed.

Figure 24. Deploy modal window



b. If **Join release** is selected as the default deployment mode, the **Join release** modal window is displayed.

Figure 25. Join release modal window



Select the type of release from the below list to join,

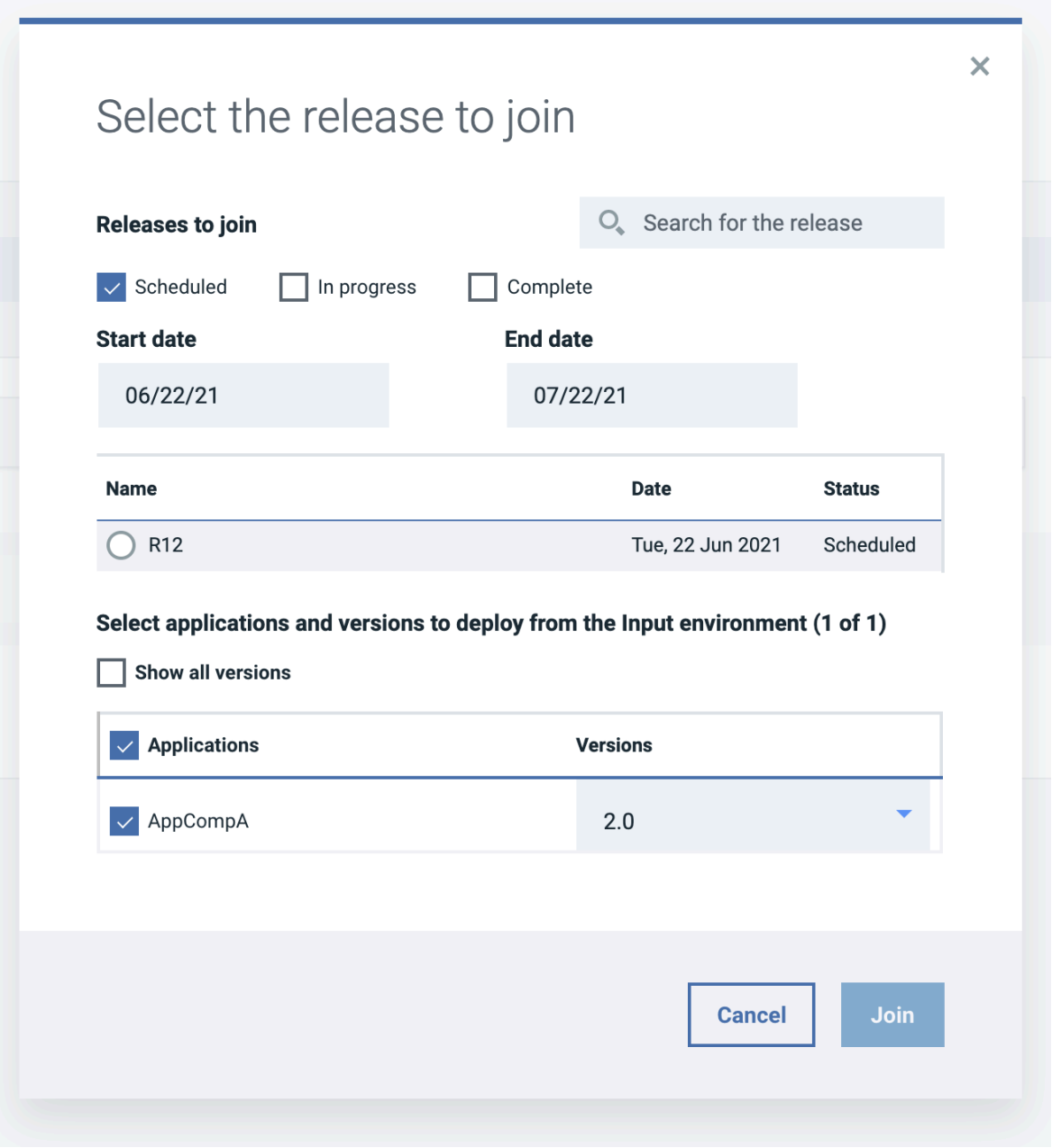
i. **Scheduled release**
ii. **In progress**
iii. **Completed**

With the release selected, select the date range to filter the available releases.

Select the required release **Name** from the list to join.

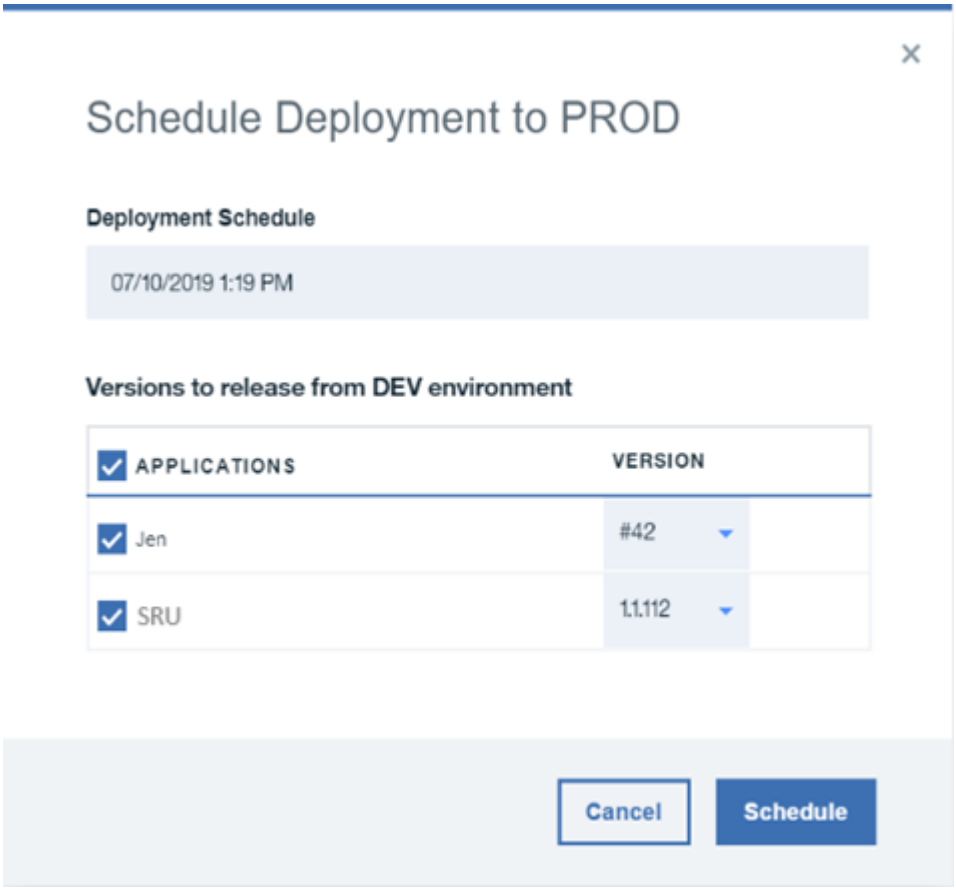
Select the **Application** and the required **Version** then click **Join** to join the release.

The below screen capture is an example for **Scheduled release** - **R12** in the date range of 06/22/2021 to 07/22/2021 for the version 2.0 of application **AppCompA**.



c. If **Schedule deployment** is selected as the default deployment mode, the **Schedule deployment** modal window is displayed.

Figure 26. Schedule deployment modal window



Example

After **Join release** is scheduled, the join release link appears below the **PROD** environment including the join release graphic and name. You can click the link to view the deployment plan page.

Figure 27. Join release link

You can use sort buttons for the alphabetical sorting of pipeline applications in ascending ( ) and descending order ( ).

What to do next

Run a deployment for a pipeline stage on page 3153.

Managing hybrid Continuous Integration (CI) and Continuous Deployment (CD) pipeline

You can create a hybrid CI and CD pipeline, add or modify the different applications and jobs to the pipeline.

About this task

Hybrid CI and CD pipeline enables you to integrate a combination of CI and CD applications to the associated deployment plan in the pipeline. Each application that you add to a pipeline is inserted onto a separate row below the last application. As you add apps and jobs to environments, tasks are added to the associated deployment plan in the order that they appear in the pipeline.

After you create the value stream, a pipeline is associated with your value stream by default. Each CI and CD application combination that you add to a pipeline is inserted onto a separate row below the last application. As you add apps and jobs to environments, tasks are added to the associated deployment plan in the order that they appear in the pipeline.

- The advanced features of hybrid CI and CD pipeline will work efficiently in Measure only when the hybrid CI/CD pipeline are not integrated with other applications.
- Whenever a user is migrating from a hybrid type environment to non-hybrid environment, ensure that all the CD physical environments that are deployed in the hybrid environment are removed. Otherwise, pipeline will not operate in the non-hybrid environment.

Limitations of hybrid CI and CD pipeline:

- The advanced features of the hybrid CI/CD pipeline will work efficiently in Measure only when the CD applications in the hybrid CI/CD pipeline are not part of any other pipeline.
- Migrating a CD application from the hybrid pipeline to the non-hybrid pipeline will not work without removing the CD application from all the logical environments in the hybrid CI/CD pipeline.

The following table provides the list of hybrid CI and CD applications currently supported in Measure:

CI application	CD application
• GitHub • Jenkins • DevOps Build	DevOps Deploy

Note: The **Full Resync** is not recommended for the applications which are part of hybrid CI and CD pipeline. You can remove the applications from the hybrid CI and CD pipeline to perform Full Resync.

1. **Optional:** On the **Pipeline** page, click **Add Environment** to create an environment.
    a. In the **Add environment** window, and in the **Environment name** field, enter the environment name.
    b. In the **Action** field, select the check boxes for the required deployment mode from the list of **Deploy**, **Join release**, and **Schedule deployment**.
    If more than one deployment mode is selected, both the default deployment mode play control and drop-down ▼ will be visible on the environment.
    The deployment mode play controls are as follows:
        - **Deploy**
        - **Join release**
        - **Schedule deployment**
    c. Based on the selected mode, in the **SET AS DEFAULT** field, select the default deployment mode.
    The deployment mode play control that is visible on the environment is the default.

d. Click **Save**.

e. Move the environment to the required location in the pipeline by hovering the mouse over the right-side of the environment, and then click and drag it to the required location in the pipeline.
You can re-order the environments except for the **Input**.

f. Click and select **Edit environment** to update the deployment mode.
In the **Edit environment** window, select the groups or users who have permission to deploy in the selected environment.

g. Click and select **Override inventory** to override the application in the deployment environment of pipeline.
In the **Override inventory** screen, select the version for the application to override deployment in the selected environment. You can use **Override inventory** in all environments except **Input**.

2. To add a CI application to the pipeline, perform the following steps:

a. To add the CI application, click **Add application**

b. On the **Add apps - Choose the apps** window, in the **Managed by** list, select a CI application.

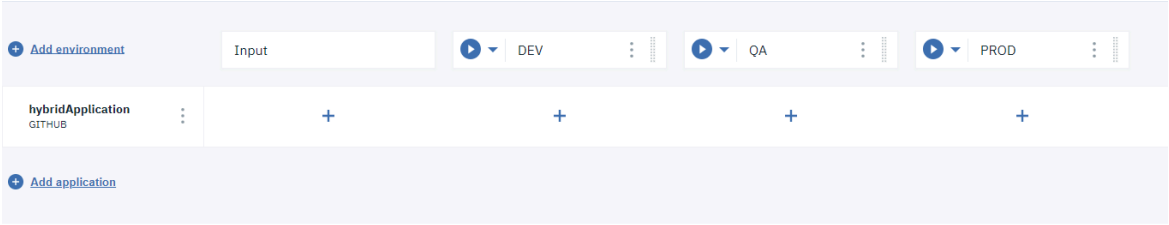
c. In the **Application name** field, enter a name for the application.

d. Click **Save**.

e. Under the **Input** environment, click to open the **Create version** window.

f. In the **Create version** window, select the required job from the list.

g. Click **Save**.



Result

You have successfully added a CI application to the pipeline.

h. In the CI application, run the build job selected in step 2.f on page 3137.

Result

You can see the build number under **INPUT**, after the build is successful in the CI application.

3. To add a CD application to the pipeline, perform the following steps:

a. Under the **Dev** environment associated with the application, click .

Note: The default environments are **DEV**, **QA**, and **PROD** environments. You can customize and create your own environments.

b. On the **Map a job to a logical environment (DEV)** window, in the **Managed by** list, select required CD application.

c. On the **Map app environments to logical environment (DEV)** window, click **Choose a different application...** > **Select your application,** choose the required application from the list, and then click **Continue**.

d. Select the process from the list, and then click **Continue**.

e. On the **Map app environments to logical environment (DEV)** window, select **dev** from the list, and then click **Save**.

Result

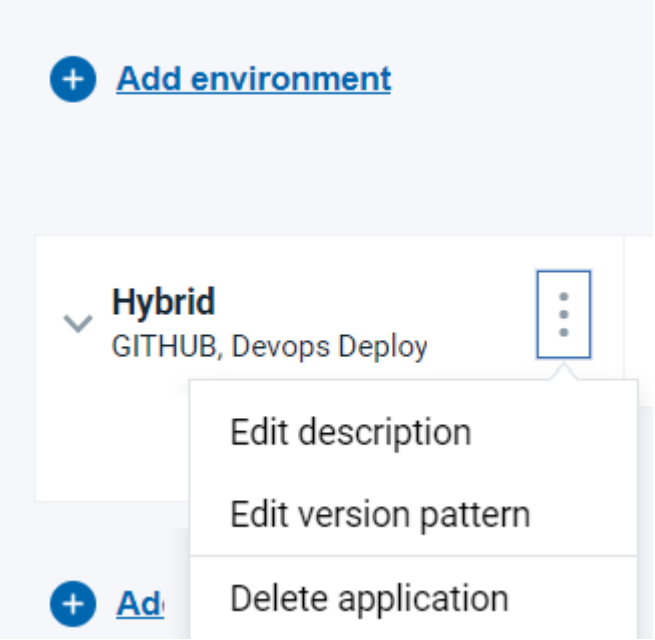
Similarly, you can map the app environments to any logical environment in the pipeline.

Result

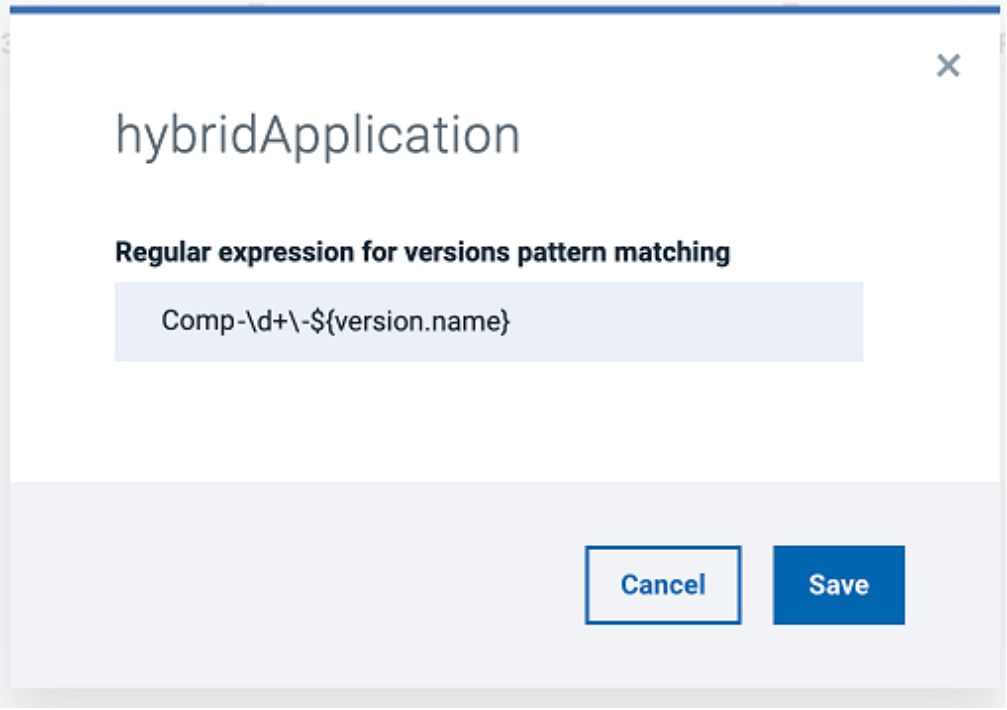
You have successfully added a CD application to the pipeline.

4. You have successfully created the hybrid CI and CD pipeline.
5. You can perform the CI and CD application versions matching by using any of the following methods:
   - **Strict matching**: To perform the strict matching of CI and CD application versions, ensure that the **snapshot name** in the CD application is same as the **build name** of the job in the CI application. Strict matching method is a default method for matching the CI and CD application versions.
   - **Regular expression matching**: To perform the regular expression matching of CI and CD application versions, perform the following steps:

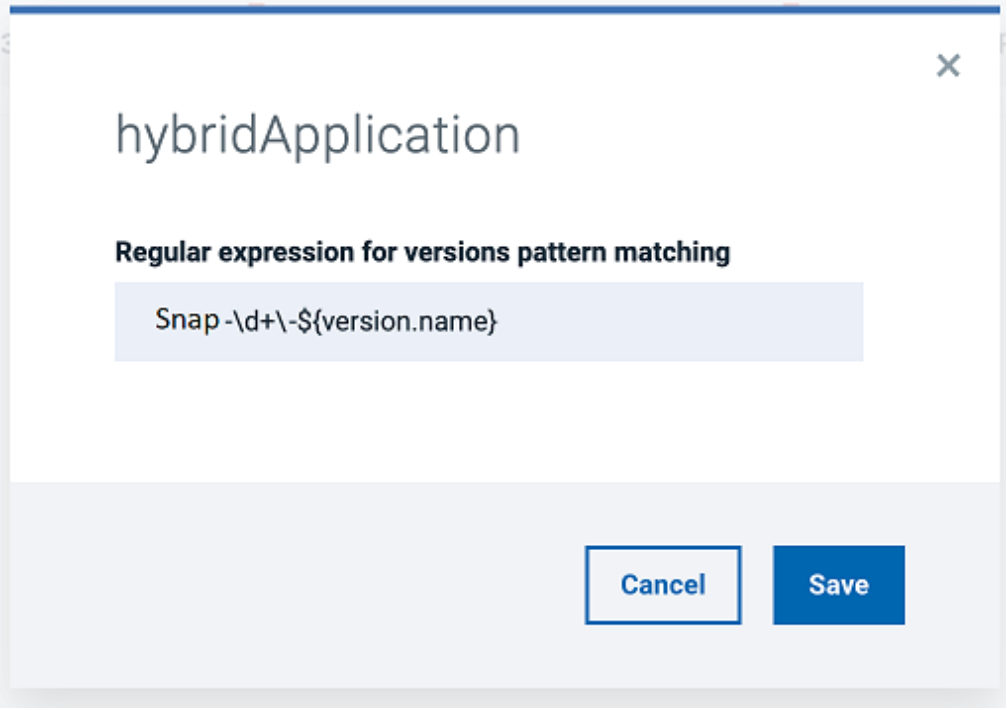
a. On the **Hybrid** CI and CD pipeline, click ⋮, and then select **Edit version pattern**.



b. On the **hybridApplication** window, perform any of the following actions to match the application versions in the CI application with the CD application.
   - **Component version pattern matching**: Add component version name for the CD application in the **Regular expression for versions pattern matching** field. Ensure that component version name of the CD application is same as the component version name of the job in the CI application.



- **Snapshot versions pattern matching**: Add snapshot version name for the CD application in the **Regular expression for versions pattern matching** field. Ensure that snapshot version name of the CD application is same as the snapshot version name of the job in the CI application.

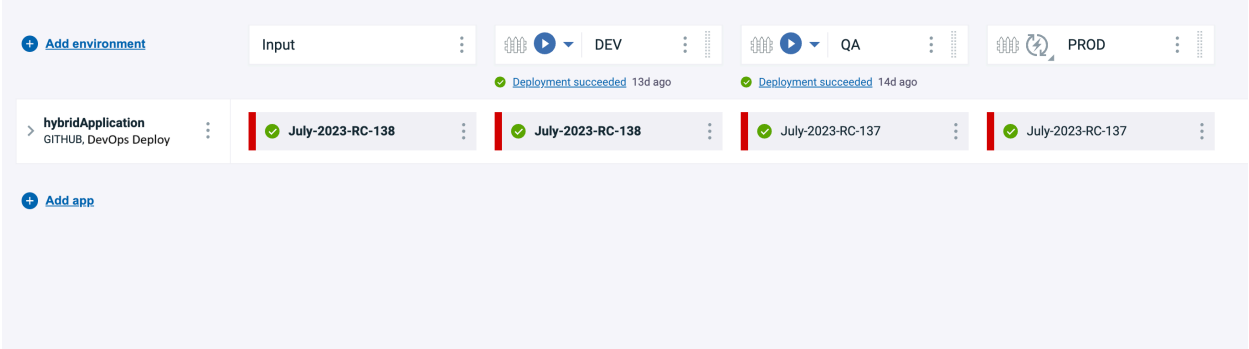


c. Click **Save**.

6. To view the deployment template associated with the environment, click ⋮ and select **View deployment template**.
   Result
   The deployment plan detail page is displayed. As you add apps and jobs to the pipeline, associated tasks are inserted into the plan. You can also add tasks directly to the plan.
7. To modify the mapped environment, click ⋮, and select **Edit application environments**.
8. To delete the environment, click ⋮ and select **Delete**.
9. **Optional:** To edit the hybrid CI and CD application components already created on the pipeline, click ⋮ on the right side of the application and select **Manage components** to add and remove components of the application, **Edit description** to modify the description of the application in the selected pipeline, or **Delete application** to delete the application in the selected pipeline.
10. **Optional:** To add a gate on page 3145, click **Add gate** and then define the gate rule.

Note: For **Automated** gates in the hybrid CI and CD pipeline, **application name** in metric input data should be **logical application name** which is defined in the hybrid CI and CD pipeline.

11. The following image is an example of the hybrid CI and CD pipeline with **DEV**, **QA**, and **PROD** environments.



What to do next

Run a deployment for a pipeline on page 3153.

Managing API-driven pipelines

API-driven pipeline applications

Before you begin

On the **Pipeline** page, you need to create a pipeline on page 3128 to incorporate API-driven applications.

About this task

API-driven pipeline applications enable you to effectively populate the pipeline and move dots through the system solely through build and deployment uploads. The create build and upload deployment API commands are displayed in the UI and can be executed in a command terminal. Advantages of using these build and deployment APIs are for providing simplified endpoints for loading data and allowing you to integrate the pipeline with unsupported tools. The relationship between the APIs and the pipeline phases that are populated are detailed in the following table.

API	Pipeline phases
builds/apiDriven	**Input**
deployments/apiDriven	**DEV**, **QA**, **PROD**, and any environments in the pipeline.

1. To map an application environment to the pipeline on the **Pipeline** page, complete the following steps:

    a. For the environment where you want to add the application, click **Add app**.

    b. On the **Add app - Choose the app** window, in the **Managed by** list, select **API-driven**.

c. Click either the **Create new application** or **Use existing appplication** radio button.

Note: The **Managed by** and **Description** fields are already populated with **API-driven**.

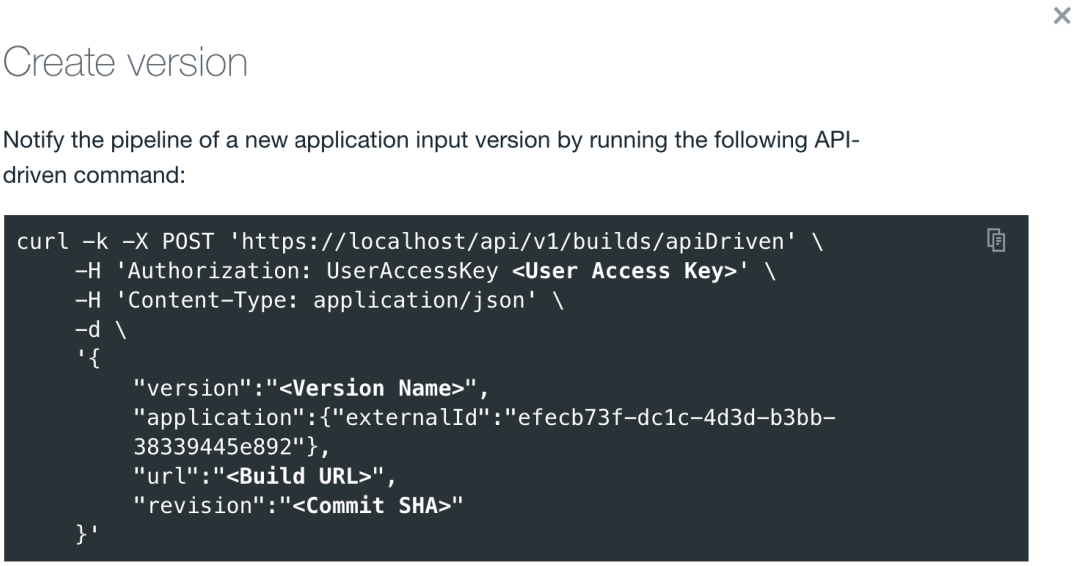
d. Based on the radio button you chose in the previous step, use the required procedure in the following table:

Create new application	Use existing application
In the **Application name** field, enter the name for the application.	This option links your pipeline application to an already existing application in Measure. You can use the API-driven REST API to create new versions and deployments for this application.
Click **Save** to add the application to the stage.	Click the **Existing application type** drop-down to list all the applications in your database and select the application. Note: If you want to use the API to create versions and run deployments for plugins without pipeline capabilities, select **Other**.
	Click the **Existing application name** drop-down to list and then select the application.
	Click **Save** to add the application to the stage.

2. To upload a build that will create a new version for the application, complete the following steps:

a. Click + under the **Input** stage of the pipeline for the **TEST API-driven Application** application, used as an example and shown in the figure below.

The **Create version** modal displays a curl command, shown in the figure below, to interact with the **TEST API-driven Application** REST API.



Note: The **Application name** entered previously in the **Add apps - Choose the apps** window is displayed in the **Create version** modal window above the `curl` command. Refer to this page on page 1891 for more details on the builds/apiDriven API.

b. Click to copy the curl command.

c. Paste the curl command in any text editor.

d. Edit the following user provided fields: `<User Access Key>`, `<Version Name>`, `<Build URL>`, and `<Commit SHA>`.

e. Copy the curl command.

f. Paste the curl command into a terminal or CI/CD automation scripts having network access to the Measure server.

Note: The displayed commands *only* work in the UNIX operating system. For MS-DOS operating systems, modifications will be required.

g. Run the curl command to update the new version of the application in **Input** environment.

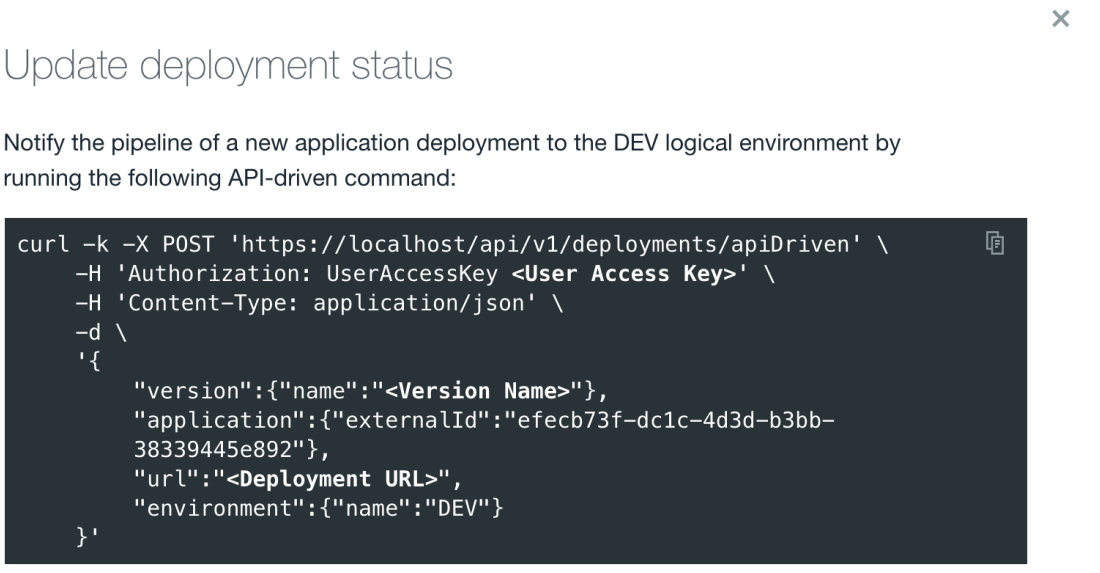
Note: Running these commands manually in a terminal are typically performed for several initial test runs. This method is *not* recommended for production use. For best practice, add the create build and upload deployment curl commands directly in the CI/CDs for automatic triggering.

Result

Succesfull run of the curl command updates the version of the application in all applicable pipelines.

3. Notify the pipeline of an **Input** version's deployment by completing the following steps:

a. Click + under the **DEV** stage of the pipeline for the **TEST API-driven Application** application, used as an example and shown in the figure of the previous step.
The **Create a deployment through the API (DEV)** modal window will display a curl command, shown in the figure below, to interact with the **TEST API-driven Application** REST API to upload a deployment to the aforementioned application.



Note: The **Application name** entered previously in the **Add apps - Choose the apps** window is displayed in the **Upload deployment status** modal window above the `curl` command. Refer to this page on page 1892 for more details on the deployments/apiDriven API.

b. Click to copy the curl command.

c. Paste the curl command in a text editor.

d. Edit the following user provided fields: `<User Access Key>`, `<Version Name>`, and `<Deployment URL>`.

e. Copy the curl command.

f. Paste the curl command into a terminal or CI/CD automation scripts having network access to the Measure server.

Note: The displayed commands *only* work in the UNIX operating system. For MS-DOS operating systems, modifications will be required.

g. Run the curl command to update the new version of the application in **DEV** environment.

Note: Running these commands manually in a terminal are typically performed for several initial test runs. This method is *not* recommended for production use. For best practice, add the create build and upload deployment curl commands directly in the CI/CDs for automatic triggering.

Result

Succesfull run of the curl command updates the version of the application in all applicable pipelines.

4. Perform the same for any required environments.

Managing pipeline gates

Gates ensure that orchestrations cannot be started in an environment until the gate rule is satisfied.

About this task

A gate is a condition that determines whether an application can run in the environment. A pipeline can have some environments with gates and some without gates. A gate condition is called a rule.

The **Gates readiness indicator** shows whether the gate rule is pass or fail for any environments in the pipeline.

In Measure 5.2.0, the gates readiness indicator verifies the status of gates rule of previous, current, and subsequent environments. There are three colors for gates readiness indicator such as Amber, Green, and Red. Each color represents the specific status of the gate rules.

Table 1167. Gates readiness indicator for 5.2.0

Gates readiness indicator	Gates rule for current and previous environment	Gates rule for subsequent environment
Amber	Fail	Pass
Green	Pass	Pass
Red	Fail or Pass	Fail

Note:

1. If there is no gates rule is set for subsequent environment, the gates readiness indicator is green for the current environment.
2. If there is no subsequent environment, the gate readiness indicator is green for the current environment.

In Measure 5.2.0, **Gates rule grouping** modal displays the gate rules and their status for each environment. There are four colors displayed for each environment based on the status of the gate rules. Each color denotes the specific status of the gate rules for each environment.

Table 1168. Gates rule status for 5.2.0

Gates rule status	Description
Amber with warning icon	Gate rules for current environment is pass but there is a gates rule failure in previous environment or approval is pending.
Green with check mark icon	Gate rules is pass for all current and previous environments.
Red with close icon	Gate rules is fail for current environment.
Gray with warning icon	Gate rules is applied but it is neither approved nor rejected.

In Measure 5.2.1 and later, the gates readiness indicator verifies the status of gates rule of the subsequent environment. There are three colors for gates readiness indicator such as Green, Grey, and Red. Each color represents the specific status of the gate rules.

Table 1169. Gates readiness indicator for 5.2.1 and later

Gates readiness indicator bar	Description
Green	Gate rules is passed or no gate rule set for subsequent environment.
Grey	Gate rule is set and it is neither approved nor rejected.
Red	Gate rule is failed in subsequent environment.

In addition to gates readiness indicator, there is an version name coloring which indicates that the previous environment failures. The version name get amber colored when there is failure in previous environment.

You can define a rule type based on the following criteria:

- **Approver** gate requires one or more responders to approve an application version before it can run in an environment. Approving a gate rule is considered passing the gate.

  Note: Anyone with access to the pipeline can create gates and be assigned as a responder. Scenarios detailing approval and rejection of application versions by responders are detailed below.
  - If a gate has multiple responders, one approval is sufficient for the application version to pass the gate.
  - If one responder rejects the application version, the application is rejected.
  - If responders both approve and reject the application version, the application is rejected.
  - When you add gates to an environment, all application versions in the affected environment must be approved before a deployment can be started. You can override a failed gate, that is, a rejected application version and you may want to do this when you run applications in a test environment.

- **Metric** gate leverages a metric-based system allowing you to automatically stop or advance application versions from being deployed into environments based on set conditions for the rule.

  Note: Using the metric rule type will provide you with visibility on the movement status of application versions and automated governance across your pipeline.

- **Compliance** gate enables you to control the deployment of application versions into different environments based on the conditions of the compliance rule.

  Note:
  - Currently under the Compliance gate, you can only create a rule for pull request approval.
  - Compliance rules are enabled only for the following type of applications:
    - Applications added by **Jenkins (Legacy)** plug-in.
    - Applications added by **Jenkins** plug-in.
    - Application added by external **HCL DevOps Deploy** and external **IBM DevOps Deploy** plug-ins and built using Jenkins. Ensure that the build details are pushed to Measure using the DevOps Deploy Jenkins publisher or pipeline plug-ins.

- **Status** gate allows you to define the statuses of a snapshot version in a Continuous Delivery (CD) application like DevOps Deploy and automatically stop or advance application versions from being deployed into different environments by matching the defined statuses with the actual statuses of the snapshot version.

  For example, when the two statuses **functional_testing** and **regression_testing** are added in the **Status** gate rule, the following scenarios illustrate when the gate will pass or fail:

Scenario 1: Status of a snapshot version in a CD application is empty.

In this scenario, the gate looks for both the **functional_testing** and **regression_testing** status in the snapshot version but the synced snapshot version will neither have **functional_testing** or **regression_testing**, so the gate will fail.

Scenario 2: Status of a snapshot version in a CD application is having one of the statuses added in the Status gate rule.

In this scenario, the gate looks for both the **functional_testing** and **regression_testing** status in the snapshot version, but the synced snapshot version will have either **functional_testing** or **regression_testing**, so the gate will fail.

Scenario 3: Status of a snapshot version in a CD application is having both the status added in the Status gate rule.

In this scenario, the gate looks for both the **functional_testing** and **regression_testing** status in the snapshot version, the synced snapshot version will have both **functional_testing** and **regression_testing**, so the gate will pass.

Scenario 4: Status of a snapshot version in a CD application is having both statuses and other additional status.

In this scenario, the gate looks for both the **functional_testing** and **regression_testing** status in the snapshot version, the synced snapshot version will have **functional_testing**, **regression_testing**, and other additional status so the gate will pass.

Scenario 5: Status of a snapshot version in a CD application is having one out of two statuses and other additional status.

In this scenario, the gate looks for both the **functional_testing** and **regression_testing** status in the snapshot version, but the synced snapshot version will have either **functional_testing** or **regression_testing**, and other additional status so the gate will fail.

Note:

- Currently the **Status** gate is not enabled for hybrid Continuous Integration/Continuous Deployment (CI/CD) pipelines. If you are using **Status** gates in hybrid CI/CD pipelines the gates will always fail.
- **Status** gate is enabled only for the applications added by the following **plug-in** versions of HCL DevOps Deploy and IBM DevOps Deploy:

- HCL DevOps Deploy 4.0.3 or later.
- IBM DevOps Deploy 4.0.3 or later.

1. For the environment where you want to add a gate, click ⋮ and select **Add gate**, and then perform the following steps.

   Note: For the environment where you want to modify a gate, click ⋮ and select **Edit gate**, and then complete the following steps.

   a. In the **Add gate** window, click **New Rule**.
   To use an existing rule, click **Existing Rules**.

   b. In the **Rule Type** field, select **Approver**, **Metric**, **Compliance**, or **Status**. Based on the **Rule Type** selected, use the following tables to add Approver, Metric, Compliance, or Status gates to environments.

Table 1170. Approver gate

"Approver" Add gate option	Value / action
Name your new rule	Enter a name for the rule.
Add Approver(s)	In the list, select responders.
Add Rule	Click to add gate to the environment, which is indicated by the **Gate** icon on the environment label. To view gate rules, click the **Gate** icon. A **Gate status** icon is added to the applications in the environment. **Note:** Initially, the **Gate status** is indicated by a vertical gray bar located to the left of the application version. If all application versions are approved, the gate is passed and the **Gate status** is a green bar. If application versions are rejected, the **Gate status** is a red bar.
Approver Rules statuses	Under the **Edit environment Gate Rules** on the right side of the **Add gate** window, you can view the statuses of the **Metric rules** and **Approver rules**. Under the **Approver rules**, select the **Send email alert to any user that requires manual approval** checkbox to receive email notification for approving the approver gates. The email notification is sent to all approver gate approvers. The approver gates approval email notification contains the **Stage Name**, **Approver Gate Name**, **Application Name**, and **Version Name** of the application for which the approval is requested.
Save	Click to populate the rule on the gate.

"Approver" Add gate option	Value / action
Send email notification to any user that requires manual approval	select the checkbox to send the e-mail notification for the designated approvers.

Table 1171. Metric gate

"Metric" Add gate option	Value / action
Name your new rule	Enter a name for the rule.
Description	Enter a description for the rule.
Metric Type	Select the required metric in the list from the following: **Coverage by Branch**, **Coverage by Function**, **Coverage by Line**, **Functional Tests**, **Static Code Analysis**, **Unit Tests**, **Container Vulnerabilities**, or **Application Vulnerabilities**. Descriptions for each **Metric Type** can be found here on page 3175.
Data Set	Select the required metric data set.
Field	Select the required field from the list. **Note:** The field is based on the **Metric Type** used for the metric rule and will be dynamically populated with selections associated with the metric. For example, if **Application Vulnerabilities** is selected, then **Blocker** will be the criteria measure for the gate.
Operator	Select the required operator from the list. **Note:** The operator is based on the **Field** that was selected and will be dynamically populated with selections suitable to the field. For example, if **Blocker** is the field, then the following list will be the available operators: **=**, **!=**, **>**, or **<**.
Value	Select the required value from the list.

"Metric" Add gate option	Value / action
	Note: The required value is entered based on the field and operator. For example, a value of zero indicating **Blocker** = 0 as the rule to pass the gate.
Occurrence period	Select the required occurrence period for the rule from the following list: **None**, **Minutes**, **Hours**, **Days**, **Weeks**, or **Months**.
Duration	Enter the duration for occurrence period of the rule.
Add Rule	Click to add gate to the environment, which is indicated by the **Gate** icon on the environment label. To view gate rules, click the **Gate** icon. A **Gate status** icon is added to the applications in the environment. **Note:** If all the application versions satisfies the added **Metric** rule, the gate is passed and the **Gate status** is a green bar. If application versions are rejected, the **Gate status** is a red bar.
Metric Rules statuses	Under the **Edit environment Gate Rules** on the right side of the **Add gate** window, you can view the statuses of the **Metric Rules**.
Save	Click to populate the rule on the gate. **Note:** For the above example of **Blocker** = 0, you may notice all the versions have a red bar indicating each had a blocker because of a failure with security scan.

Table 1172. Compliance gate

"Compliance" Add gate option	Value / action
Name your new rule	Enter a name for the rule.
Description	Enter a description for the rule.
Resource Type	Select the resource type from the list.
Field	Select the required field from the list.
Operator	Select the required operator from the list.

"Compliance" Add gate option	Value / action
	Note: The operator is based on the **Field** that was selected and will be dynamically populated with selections suitable to the field.
Value	Select the required value from the list. **Note:** The required value is entered based on the field and operator.
Add Rule	Click to add gate to the environment, which is indicated by the **Gate** icon on the environment label. To view gate rules, click the **Gate** icon. A **Gate status** icon is added to the applications in the environment. **Note:** If all the application versions satisfies the added **Compliance** rule, the gate is passed and the **Gate status** is a green bar. If application versions are rejected, the **Gate status** is a red bar.
Compliance Rules statuses	Under the **Edit environment Gate Rules** on the right side of the **Add gate** window, you can view the statuses of the **Compliance Rules**.
Save	Click to populate the rule on the gate.

Table 1173. Status gate

"Status" Add gate option	Value / action
Name your new rule	Enter a name for the rule.
Description	Enter a description for the rule.
Status (Comma Separated List)	Add the statuses which you want to match with the snapshot version in a CD application. You can add multiple statuses by separating them with a comma. Note: status names are case sensitive, enter the exact name in the snapshot version, if not the gates will fail.
Add Rule	Click to add gate to the environment, which is indicated by the **Gate** icon on the environment label. To view gate rules, click the **Gate** icon. A **Gate status** icon is added to the applications in the environment. **Note:** If all the status defined in the status rule matches the status added in the snapshot version, the gate is passed and

"Status" Add gate option	Value / action
	Gate status is a green bar. If the statuses does not match the gates are failed, the **Gate status** is a red bar.
Status Rules statuses	Under the **Edit environment Gate Rules** on the right side of the **Add gate** window, you can view the statuses of the **Status Rules**.
Save	Click to populate the rule on the gate.

c. In the **Add gate** window, on the right side, you can hover over a rule and click **Remove rule** to delete the individual rules.

2. In the **Environment** stage, click the **Gate** icon to view the gate rules.
   Result
   You can see the number of rules and approvers for each gate.
3. In the **Pipeline** window, click the **Gate status** icon for the application version with the gate to open **Version Rules** window.
   Result
   You can see the status of the gates and designated approvers can approve or reject the approver gate rule.
4. On the **Version Rules** window, click **Approve** or **Reject** and then click **Save** to approve or reject the approver gate rule.

   Note: If you are an approver for multiple rules, you can approve all or some of them and reject others. If you are not a designated responder, you can neither approve nor reject the gate rule.

5. For the environment where you want to delete a gate, click and select **Delete gate**.

What to do next

Run a deployment for a pipeline stage on page 3153.

Running pipeline applications

Run applications in a pipeline environment.

About this task

You can manually run an on-demand deployment, schedule a deployment for a later time, or join a deployment to a release. When you start a deployment, the deployment plan template associated with the environment is used. Tasks for applications in the environment are run in order starting with the first task defined in your deployment plan template. You can view the deployment's progress by using the supplied link.

1. To run a deployment, complete the following steps:

    a. On the **Pipeline detail** page, click the environment's **Deploy** button.
    The **Deploy** button is available if you have sufficient permissions to run applications in the environment.

    b. In the **Deploy to** window, select the applications that you want to run in the environment.
    By default, all applications are selected in which the latest version in the previous environment has not yet been deployed.

    c. In the **Version** column, select the version.
    If an environment has a gate rule assigned to it on page 3145, only application versions that pass the gate rule can deploy. Versions that have passed the gate rule have the status of **Passed**.

    Note: For hybrid CI/CD pipelines, only the matched versions will be available for deployment.

    d. If a gated environment has an unapproved application version, approve the version.
    Only users who are designated responders can approve or reject application versions.

    Note: You can override rejected application versions. To deploy a rejected version, confirm that you intend to override the gate and run the deployment anyway. To override a gate, you must be assigned the Lead Release Manager or Release Manager roles.

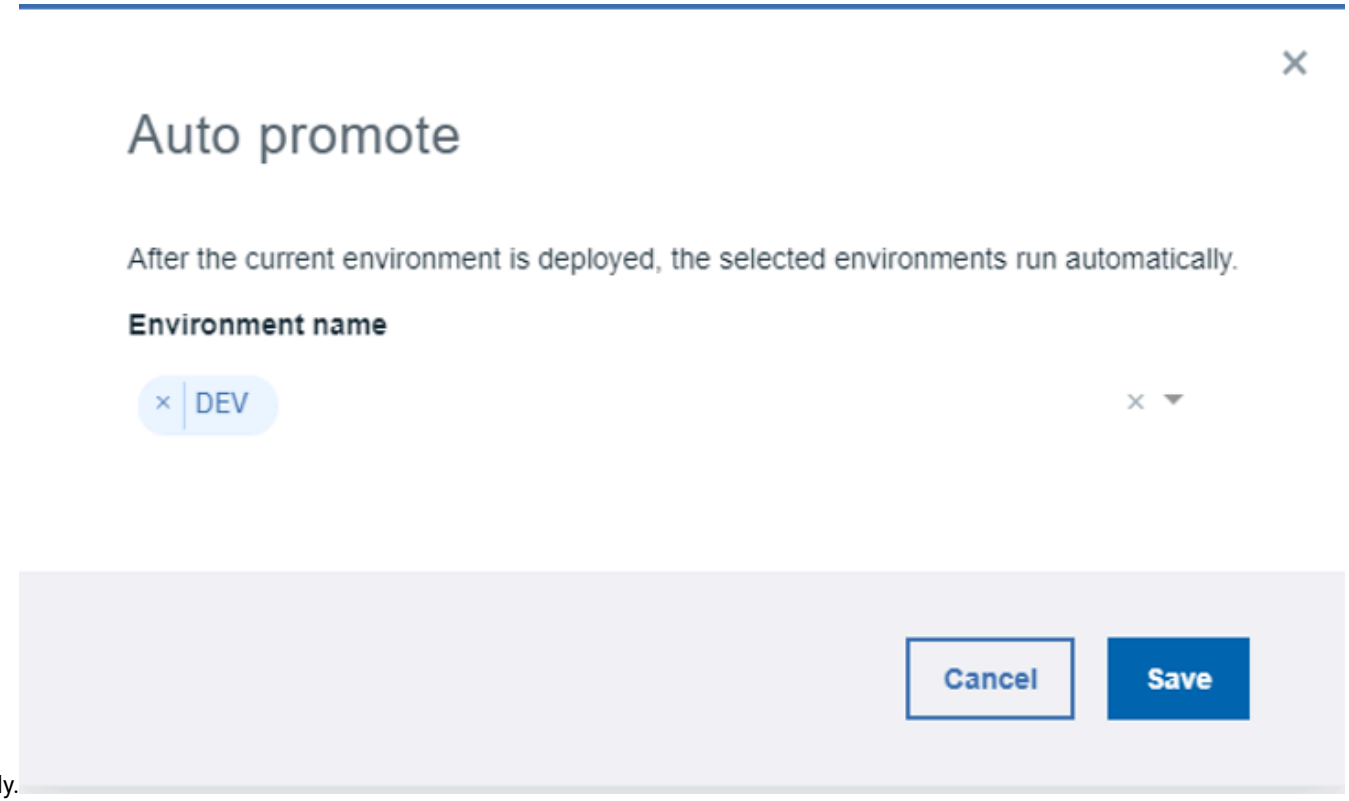
    e. Click **Deploy**, to run the selected application versions.
    Result
    A deployment starts with the deployment plan associated with the environment.

2. **Optional:** To schedule a deployment for a stage, click **Schedule deployment** on the context menu for the stage, and then select a future date and start time.
    A deployment for the stage deployment plan will start at the scheduled time. Pipeline scheduled deployments are displayed on the calendar.
3. **Optional:** To join a deployment to a release, click Join release on the context menu for the stage, select an associated release, and then a application version. The release must have been preemptively created. The deployment plan will start based on the scheduled start time of the selected release.
    If you would like to start it earlier, you can start the deployment by manually starting one of the eligible deployment plan tasks, or click **Edit this plan** on the deployment plan in the release. As with other deployment plans, you can add and delete tasks on the pipeline-generated plan at any time. On the release or plan page, use the **Go to pipeline...** link to open the associated pipeline or stage.

What to do next

You can auto-promote DevOps Deploy and Jenkins application from input stage to any other stage environments. To auto-promote, click ⋮ icon next to **Input** and then select **Auto promote**. In the following modal, select or deselect the environment(s) you wish to promote your version.

The auto-promote process will consider your gating criteria when determining whether to deploy the new version automatically.



Note: You can deploy the application manually in the auto-promotion environment. Click icon in stage and then select **Deploy** to deploy the application manually.

Running open pipeline deployment

You can run the deployment and release of the application using the Measure pipeline.

To run a deployment from Velocity pipeline perform the following steps,

1. Create a pipeline for the project in plugin.
2. Link the project to the repository in plugin.
3. Integrate the plugin by following the steps in integration section.
4. Add plugin project to the environment of Measure pipeline by following the steps in managing pipeline section on page 3128.
5. The latest build version of the plugin project gets populated in the **Input** stage of the Measure pipeline.

6. To deploy the application, click for the environment and select the required version of the project for deployment.
7. Click **Deployment from plan** to view the deployment task status. After the completion of deployment task, the status changes to **Complete**.
8. The result of the project deployment is updated in pipeline.

Solution versions

Solution version is a combination of different versions of one or more applications in a pipeline.

You can use a solution version to combine the required application versions from the pipeline and deploy them to different stages. A solution version includes the application versions that are tested and working in a stage and can be deployed into any stage in the pipeline. You can roll back or upgrade the version of applications in the solution version. You can add only the compatible combinations of application versions to the solution version.

The solution version is built incrementally to ensure that the tested combinations are included. You can create only one solution versions with a combination of deployed applications versions in a stage. You can deploy the solution version created in a stage into other stages in the pipeline.

During the build phase, a solution version is called a solution draft version. Any changes to the deployed application versions of a draft solution version in a stage are updated automatically in the solution version inventory. You can edit the combination of applications by editing the draft solution version. After the solution version is published, the version of the solution version is locked and its inventory cannot be updated.

When a new solution version is deployed in the stage where there is already a solution version is deployed, then the published solution version is either reverted to draft status or removed completely.

Enabling the solution version feature on Measure

When you want to use the solution version on Measure, you must enable it.

1. Log in to Measure.
2. Navigate to the following URL:

   `https://hostname:port/featureflags`

   Result

   The **Feature Flag** page is displayed.
3. In **Enter a feature name to enable it** field, enter `pipeline-solution-versions`.
4. Click **Enable**.

Results

You have enabled the solution version feature on Measure.

Adding an application to the pipeline

You can add an application to the pipeline in Measure.

Before you begin

You must have created a default or any customised value stream in Measure.

1. Perform the following steps to add an application to the pipeline:
    a. In the Measure home page, on the left navigational panel, click .
    Result
    The value streams that you created are displayed.
    b. Select the value stream that you created from the list.
    c. Click **Pipeline** on the panel.
    Result
    The **Pipeline** page is displayed for the value stream.
    d. Click **Add application**.
    Result
    The **Add app - Choose the app** window is displayed.
    e. In **Managed by** list, select application.
    f. In **Application name** field, enter a name for the application.
    g. Optional: In **Description** field, enter a description for the application.
    h. Click **Save**.
2. Perform the following steps to create a version for the application in the pipeline:
    a. Click **Create version** under the **Input** column.
    Result
    The **Create version** window is displayed.
    b. Select the required job from the list, and then click **Save**.
3. Perform the following steps to map the application to **DEV** environment:
    a. Click under the **DEV** column.
    Result
    The **Map a Jenkins job to a logical environment (DEV)** window is displayed.
    b. Select the required job from the list, and then click **Next**.

c. Specify a build number or select **${version.buildNumber}** from the **buildNumber** list. and the click **Next**.

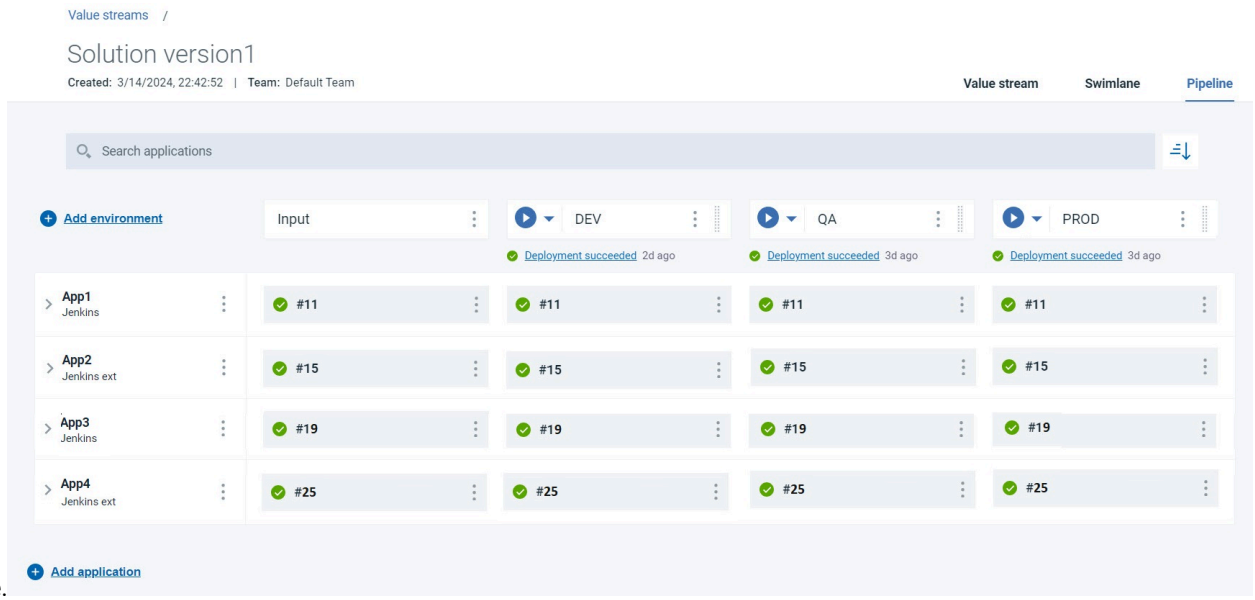
d. In **Select Version Parameter** list, select **buildNumber**.

e. Click **Save**.

Result

Similarly, you can map the app environments to any logical environment in the pipeline.

4. If required, do the above steps again to add the applications and map them to the pipeline.



Results

You have added an application to the pipeline.

Creating the solution version

You can create a solution version in the pipeline stage with combination of application versions.

Before you begin

You must have completed the following tasks:

- Enabled `pipeline-solution-versions` option in feature flags page.
- Created a pipeline with one or more applications.
- Deployed the applications in the pipeline.

About this task

You can create a solution version with a combination of the applications version in the pipeline.

1. Perform the following steps to navigate the required pipeline page:
    a. In Measure home page, on the left navigational panel, click .
    b. Select the required value stream from the list.
    c. Click **Pipeline**.

    Result

    The **Pipeline** page is displayed.
2. Perform the following steps to create a solution version:
    a. On the **DEV** environment, click , and then select **Solution Version > Create**.

    The **Create Solution Version** window is displayed.

    Note: An error message **At least one application version must be there to create solution version** will be displayed if application version is not created for the application.

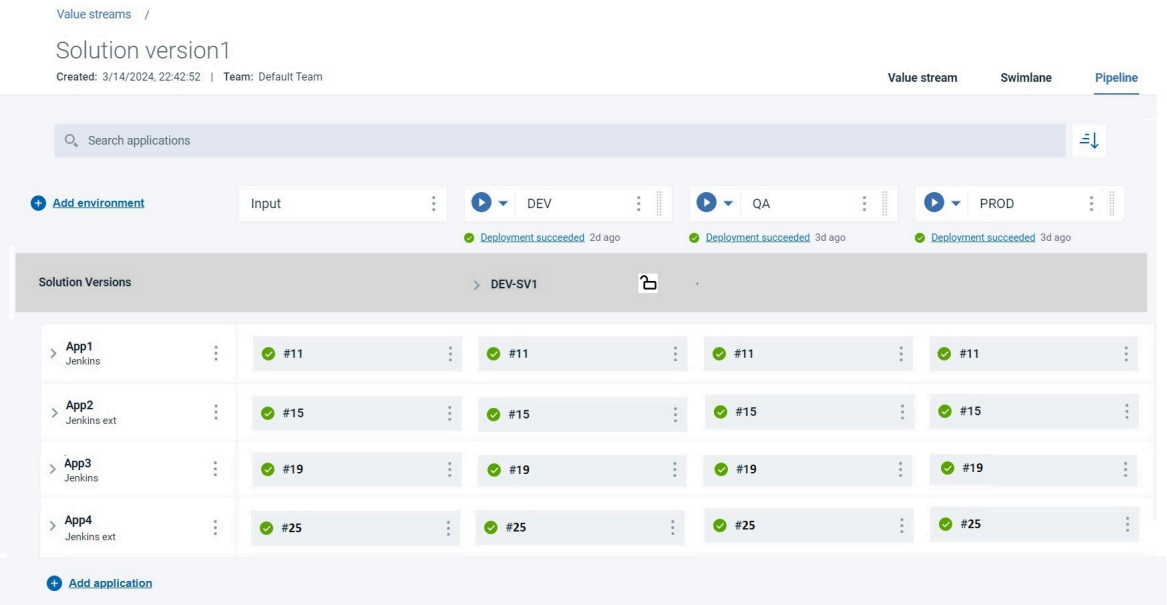
    b. In the **Name** field, enter a name for the solution version.
    c. Optional: In the **Description** field, enter a description for the solution version.
    d. In the **Versions** list, select the Application versions based on the requirement.

    For example, if you have added four applications such as **APP1**, **APP2**, **APP3**, and **APP4** to the pipeline, the **Versions** filter displays the **APP1**, **APP2**, **APP3**, and **APP4** with the version numbers.

    You can select the required applications version from the list when you create the solution version.
    e. Click **Create**.

    The **Solution Versions** row is displayed below the **Environment** column.

    The draft solution version is created for **DEV** environment. The draft solution version is indicated with icon.

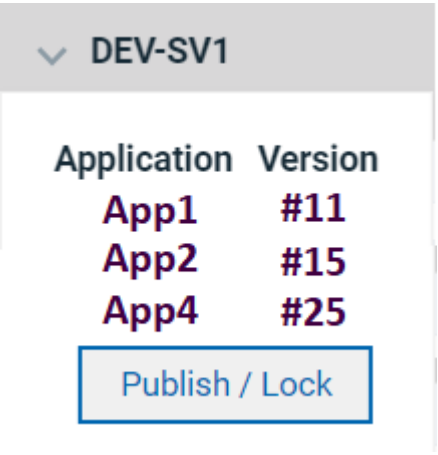


You can create the solution version for all the stages in the pipeline except for the **Input** phase.

Note: Only the deployed solution versions are displayed in the **Solution Versions** row in the pipeline page. You can create a solution version with different combination of application versions and deploy them in any stage. You can create one solution version It is not allowed to create multiple solution versions for a particular stage.

f. To view the application versions linked to solution version, click icon on the solution version .

A window is displayed with the application versions liked to the solution version.



The above example shows that the **App1**, **App2**, and **App4** were added to the solution version **DEV-S1**.

Results

You have created a draft solution version in the pipeline stage.

Related information

Managing pipeline gates on page 3145

Editing the draft solution version

You can edit the draft solution version to update name, description, and application version combinations.

About this task

You can edit the draft solution version at all the stages in the pipeline. It allows you to update the name and select the required application versions. The draft solution version is indicated by icon. It is not possible to edit the solution versions that are locked or published.

1. Click , and then select **Solution Version > Edit**.
   Result

   The **Edit Solution Version** is displayed.
2. You can perform the following actions:
   - Modify the name and description for the solution version.
   - Choose the application versions from **Versions** list.
3. Click **Save**.

Results

You have edited the draft solution version.

Publishing the draft solution version

You can publish the draft solution version in the pipeline stage to lock the application versions.

About this task

You can publish the draft solution version at all the stages in the pipeline to lock the application versions. The published solution version is indicated by icon. After publishing, you cannot make any changes to the solution version.

1. You can perform any of the following actions to **Publish** the draft Solution version:
   a. Click on the draft solution version, and then select **Publish / Lock**.
   b. Click on the environment, and then select **Solution Version > Publish**.

      The **Publish** window is displayed.

2. Click **Confirm**.
   Result
   The icon is suffixed to the published solution version on the **Solutions Versions** row.

Results
You have published the solution version in the pipeline.

Deploying the solution version

You can deploy the solution versions in the environments.

About this task
You can deploy the draft solution version and published solution version in the pipeline. When a new solution version is deployed to the environment where the published solution already exists, then the published solution version is either reverted to draft status or removed completely.

1. On the pipeline stage, you want to deploy the solution version, click , and then select **Deploy**.
   Result
   The **Deploy** window is displayed.
2. Select **Solution Version** tab.
3. Select the solution version from the list for deployment.

   Note: You can deploy all the **draft** and **published** solution version at all stages in the pipeline.

4. Click **Deploy**.

Results
You have deployed the solution version in the pipeline.

Unlinking the solution version

You can unlink the solution version from the pipeline stage.

About this task
You can unlink the draft and published solution versions in the pipeline. The unlinked solution versions are not deleted from the pipeline. You can also add the unlinked solution versions to the draft solution versions.

1. On the pipeline stage, click , and then select **Solution Version > Unlink Solution Version**.
2. Click **Confirm**.
   Result
   Solution version is unlinked from the environment.

The unlinked solution version are not permanently deleted from the stage. To reuse the unlinked solution version, .
You can find the unlinked solution versions under the **Solutions Version** tab.

3. To view the unlinked solution versions, click icon on the stage.
   Result
   The unlinked solution versions are listed under the **Solution Version** tab.

Results

You have unlinked the solution version from the pipeline.

Deleting the solution version

You can delete the solution version permanently from the pipeline stage when it is not required.

1. On the pipeline stage, click , and then select **Solution Version > Delete**.
   Result
   The **Confirm delete** is displayed
2. Enter the solution version number in the field, and then select **Delete**.
   Result
   The solution version is permanently deleted from the pipeline.

Results

You have deleted a solution version permanently from the pipeline stage.

Data Insights

The Insights feature provides DevOps metrics for your project data.

The Measure metric data tools enable you to assess your projects' efficiency and quality. You can measure every aspect of the development lifecycle with the supplied charts. You can create your own charts with metric definitions and upload custom data to Measure using the API endpoints on page 3091. Because Velocity data can come from plugins and API calls, project data can come from virtually anywhere, including planing and development tools, testing and build applications, and deployment processes.

The Insights feature consists of three focus areas:

Analyze project data with charts and graphs and then save the information in Insights dashboards on page 3165.

The HCL DevOps Deploy (Deploy) reports feature provides reports for data imported from Deploy.

Note: This feature will be deprecated in a future version.

The experimental Portfolio feature displays metric data for applications.

Managing Insights dashboards

Managing Insights dashboards

Managing Insights dashboards

Use the Insights dashboard feature to analyze project quality and efficiency.

Build dashboards by selecting from a large variety of charts. You can customize and annotate charts and download the data sets.

Dashboards are containers for customized metric data on page 3165. You select data by applying filters and queries to project areas, such as value streams. After you define data scope, attach charts and graphs to the dashboard. Measure provides a large variety of pre-defined charts, and you can create your own custom metric definitions on page 3091. Dashboards record your selections and automatically apply the same filters and graphs the next time you open them. You can create multiple dashboards, and each one can have unique criteria. If your organization has numerous teams, for example, you might create a separate dashboard for each one. The available charts are described later in this section on page 3175.

Additional managing Insights dashboards details are included in the following topics:

Creating dashboards

Create dashboards and favorites to manage metric data charts and graphs.

About this task

You create dashboards to by attaching charts to your project data that help you analyze your projects quality and efficiency.

Dashboards are created on the **Insights Dashboards** page. After you create a dashboard and customize it with charts and filter conditions, you can save and use for future sessions. The **Delivery Performance** dashboard is provided by default. It has several charts preconfigured, including the **Lead time**, **Cycle time**, and **Build count** charts. You can customize the **Delivery Performance** dashboard by adding new charts, and setting filters and data scope conditions.

To create a dashboard, complete the following steps:

1. On the Measure **Home** page, click **Insights > All Dashboards > Create dashboard**.
2. In the **Create dashboard** dialog, in the **Name** field, enter a dashboard name.
3. In the **Description** field, enter a distinctive description for the dashboard.
4. Click **Add**.

   Result

   Dashboards are also displayed either in card view by clicking or in list view by clicking on the **Insights All Dashboards** page. In the list view, you can sort fields in ascending or descending order by clicking on the following field names:

- **Name**
- **Description**
- **Creator**
- **Last Modified**
- **Modified By**

Each card or list view item contains status information about the dashboard, and provides a link to the dashboard details. In either list or card view, you can click and then select **Delete** to delete the dashboard.

Note: You can use Ctrl+P to generate the portable document format file of dashboard reports.

5. To add a created dashboard to your favorites, click the **All Dashboards** tab and then click to display the list view of dashboards. In the list, locate the **Name** of dashboard to be added to your favorites and then click in the left side of the field.
   The dashboard is listed on the **My Favorites** tab.
6. To remove a dashboard from your favorites, click and the dashboard is removed from the **My Favorites** tab.
7. To remove a dashboard from the **My Favorites** tab in card layout, click and then click for the dashboard to remove and select **Unfavorite**.
8. To remove a dashboard from the **My Favorites** tab in list layout, click and then click for the dashboard to remove and select **Unfavorite**.

What to do next

Customize the dashboard by filtering project data and applying charts to the data on page 3165.

Analyzing Insights data

Managing Insights dashboards.

About this task

With Insights dashboards, you measure your organization's progress and efficiency by applying analytic tools to your project data. Data comes from value streams, pipelines, HCL DevOps Deploy (Deploy) applications, and other project areas. By default, all data related to the teams to which you belong are available. You set data scope by selecting built-in filters. You can set the scope by application, team, value stream, or some combination.

You can add charts to the dashboard from a large collection of DevOps chart types. Chart data is determined by the metric definition associated with a chart on page 3170. When you add a chart to the dashboard, its graph area fills to reflect the filtered data. As you add multiple charts to a dashboard, the charts rearrange themselves to accommodate the new chart. When you save the dashboard, Measure saves the filter and the charts that you used.

The dashboard workflow follows these general steps:

1. Set the scope by defining a filter. You can scope data by teams, value streams, or applications.
2. Set the search limits.
3. Add a chart to the dashboard.
4. Customize the chart. You can change the chart style for all chart types. You can apply additional filters to some chart types, such as Deploy Count.

To manage a dashboard, complete the following steps:

1. On the Measure **Home** page, click **Insights > Dashboards**, and then select the dashboard.
   Result
   The dashboard page opens, which enables you to filter data, and apply charts.
2. In the **Dashboard** page, define the data scope by completing the following steps:

   a. In the **Filters** area, select one or more filters.
      By default, the data for all teams to which you belong populate the charts. You can restrict the data to a specific scope by selecting filters. Filters are available for teams, value streams, and applications. For example, you can restrict data to a specific team by selecting the team. Then the charts that you add will only contain data for the selected team. You can select more than one filter. Measure teams and teams integrated from Deploy are available.

   b. With the **Calendar** drop-down list to the right of the calendar, , set the search period.
      The default search boundary is **Last 90 days** or you can also select from one of the following: **Today**, **Last 24 hours**, **Last 7 days**, or **Last 30 days**.

   c. With the drop-down list to the right of **Group By**, set the x-axis coordinates for the charts.
      The default x-axis coordinate is **Day** or you can also select from one of the following: **Week**, **Month**, **Quarter**, or **Year**.

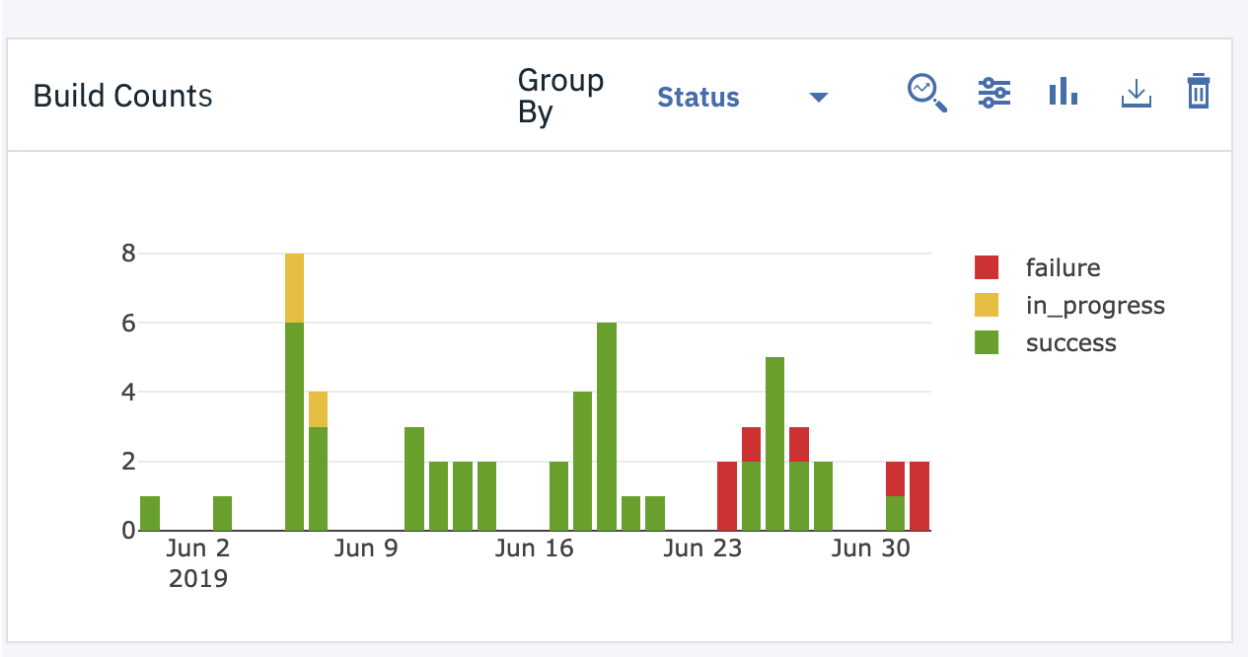
3. Click **Add charts** area and select a chart.
   The chart is displayed on the dashboard. If the dashboard already contains charts, the new one is inserted in the top row. Data that matches the filter condition populates the chart.

   Note: Depending on the filter and chart type, a chart might not contain significant data.

4. **Optional:** On the chart, use the **Group By** drop-down to set the data grouping.
   Result
   The **Group By** function modifies the data presentation without changing the data.

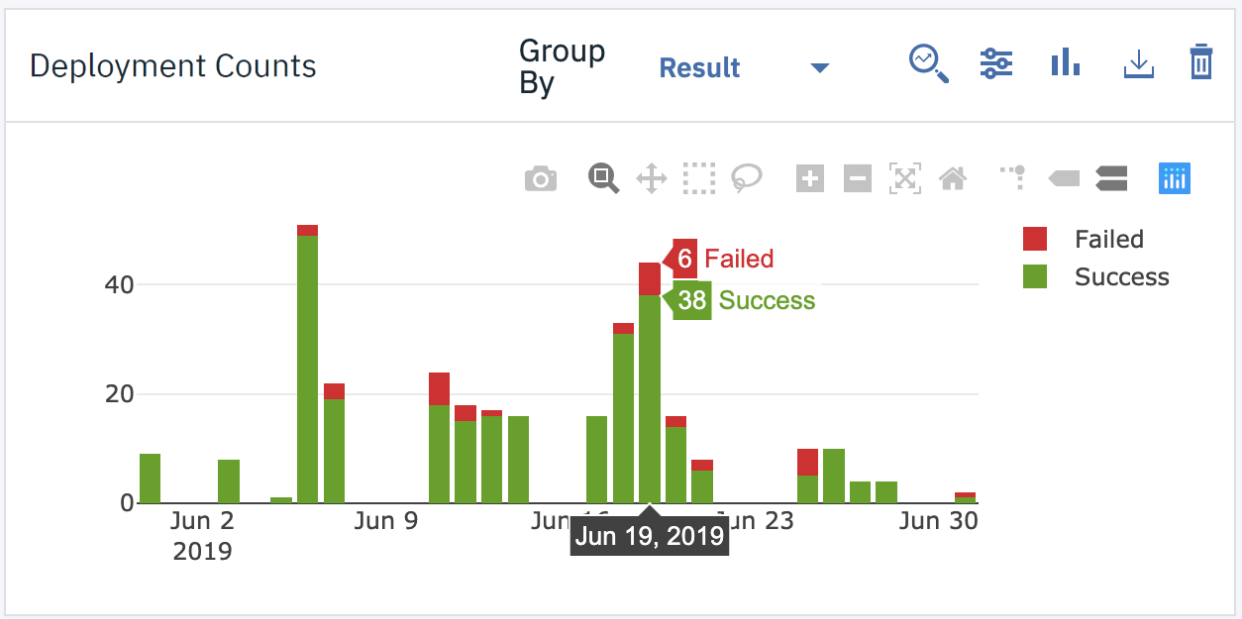
   The following illustration shows a delivery flow chart, **Build Counts**. Each bar in the graph represents the number of software builds for the specified coordinate, in this case days. The chart's **Group By** function is set to **Status**. A chart's **Group By** filter is determined by the metric definition assigned to the chart and might be unique to the chart. The chart's key displays the values for this grouping. This particular chart can also be grouped by **Name**, **Requestor**, **Application**, or **Tag**.

Note: Some chart types provide a default grouping that cannot be changed.



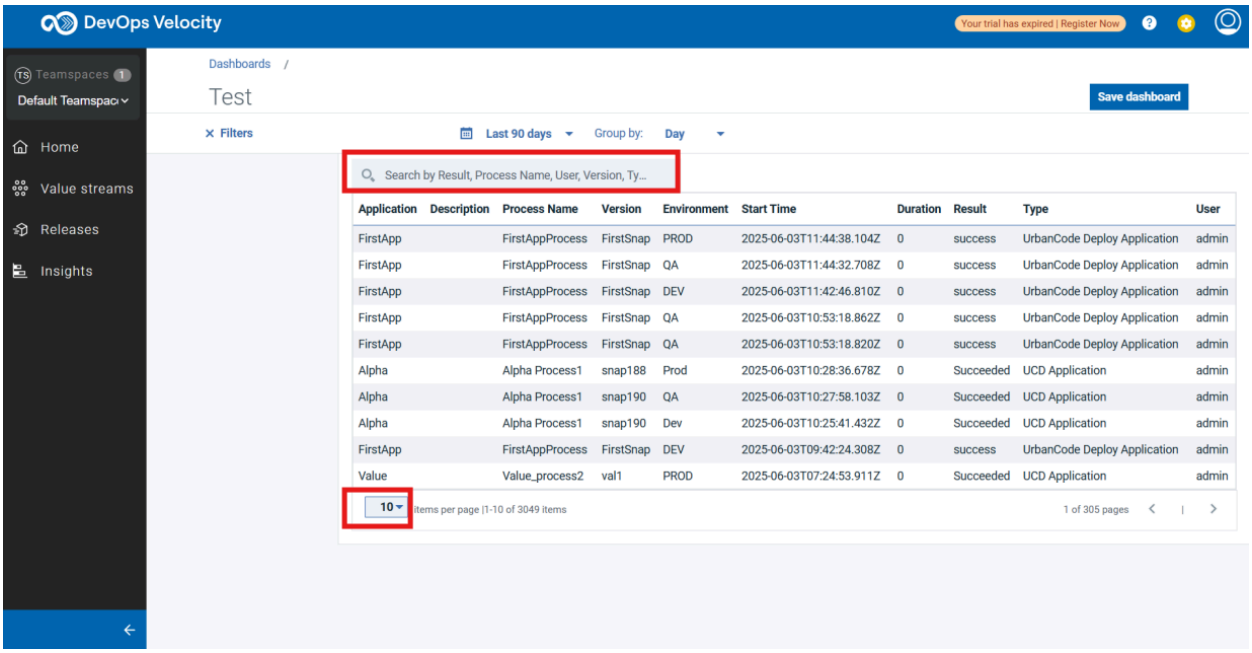
To see the breakdown of values for a grouping, hover the mouse over the interactive graph element. The following illustration shows another delivery flow chart, **Deploy Count**. This chart is grouped by result. Hovering the mouse over a bar shows the results for the selected coordinate.

Note: For the **Deployment Counts** chart only, the choices in the **Group By** list are: **Process name**, **Environment**, **Result**, **Type**, **User**, **Version**, **Application**, **Tag**, or **Team**.



5. **Optional:** On the chart, click icon to view the dataset grouping in table format.
   Result

   The following illustration shows a delivery performance chart, **Build Counts** in table format.

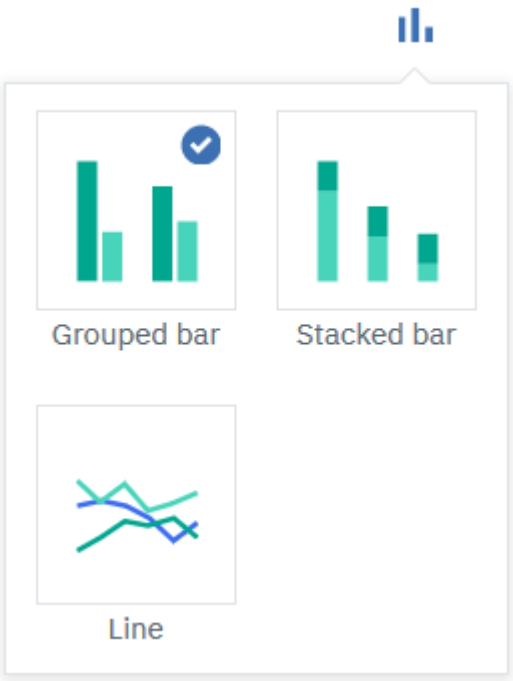


You can enter any keyword or phrase of the field listed in the **Search** bar to find the required data in the dataset table. You can use the pagination feature to define the number of items to display in the dataset table. The data in the table can be sorted in ascending and descending order.

6. **Optional:** On the chart, click the drop-down to refine the data filter.
   Result
   The chart is filtered, and the key is updated. The chart filter refines the data scope by applying additional filters to those you selected earlier.

   Note: Not all chart types provide a chart-specific filter. For the **Deployment Counts** chart only, the choices in the **Filter** list are: **Data set**, **Environment**, **Process name**, **Result**, **User**, or **Version**.

7. **Optional:** On the chart, click the drop-down to change the chart type.
   The types available are **Grouped bar**, **Stacked bar**, and **Line**, as shown in the following illustration. The default type is **Grouped bar**.



8. **Optional:** On the chart, click , to download a comma-separated-values file of the chart's data.
9. Click **Save Dashboard**.
   Result
   Measure saves your filters, charts, and chart customizations.
10. Use Ctrl+P to generate the portable document format file of dashboard report.

What to do next

You can create your own metric definitions and upload custom data on page 3091 to Measure. Custom charts are displayed in the **Add charts** area along with the pre-defined charts.

Metric chart reference

Metric charts can be attached to dashboards, value streams, and applications in the Portfolio view.

Measure provides metric definitions that you can apply to value streams, deployments, pipelines, and applications. Metric definitions describe expected data types and default chart settings. You can also send custom metric definitions to Measure from the external clients by using the API on page 1893.

Metric definitions, when applied to data, are represented by charts and graphs that can be attached to dashboards, value streams, and applications in the Portfolio view. Depending on how the data is collected, metric charts can be updated in near-real-time.

Measure provides many pre-defined metric definitions and their related charts. You can create your own custom metric definitions and use them wherever you would use a pre-defined one. on page 3091

Delivery flow charts

Table 1174. Delivery flow-type charts

Chart	Description
Lead Time	The time required for a dot to go from business acceptance to value realization. This includes both engineering time and non-value-added waiting time between sub-processes.
Lead Time Key Performance Indicator (KPI)	The Lead Time KPI measures the efficiency of a dot to reach the defined target. You can define the target using the status rule. To define target for lead time KPI, see here on page 3173.
Cycle Time	This is similar to lead time, except cycle time is measured from when the development team begins work to when that work reaches the customer.
Cycle Time Key Performance Indicator (KPI)	The Cycle Time KPI measures the efficiency of a dot to reach the defined target. You can define the target using the status rule. To define target for cycle time KPI, see here on page 3172.
Dev Cycle Time	This is similar to cycle time, except dev-cycle time is measured from when the development team begins work to when that work reaches end of the development (before the production stage).
Build Counts	Number of builds by type in a specified time period.
Deployment Counts	Number of deploys by status in a specified time period. The metric can be set to process names, environment, result, type, or application. The default metric is result. The data can scoped by data set, environment, process, result, or version.
Load	The number of work items active or waiting in a value stream at a given time. Load measures utilization capabilities of value streams related to productivity in the process flow. "Active" refers to the stages from the lead-time start until the lead-time end.
Throughput	The rate of work items completed during a period of time. Improving throughput can result in better responsiveness to customer requirements and may yield lead time reductions for value streams.

Table 1174. Delivery flow-type charts (continued)

Chart	Description
Agile Churn	The ratio of the sum of the values of story points added and removed to the value of the planned story points of a sprint. It calculates the percentage of the change in planned scope for any sprint.
Distribution	The proportion of different types of work items over time. This provides teams visibility into the type of work being completed (features, defects, tasks, and so forth.)
Flow Efficiency	The ratio of actual development time to cycle time. It is expressed as a percentage, which helps teams to understand their performance.
P.R.E Risk	Probabilistic Risk Estimator (P.R.E) Risk provides estimated probability to miss current sprint goal.
P.R.E Deadline	P.R.E Deadline provides estimated complete date for the current sprint.

DevOps Research and Assessment (DORA) metrics

Table 1175. DORA charts

Chart	Description
Deployment Frequency	The number deployments into production environment.
Lead Time to Change	The time required for a dot to go from commit to production deployment.
Time To Restore Service (MTTR)	The time taken to resolve the incident.
Change Failure Rate (CFR)	The ratio of the number of failed changes to the total number of changes deployed within a specified time period. It is expressed as a percentage, helping teams understand the rate at which their deployments are successful or failed.

Cycle Time KPI

To add **Cycle Time KPI** chart to dashboard, complete the following steps:

1. On the navigation bar, click **Insights**.
2. Select the required dashboard.
3. Under the dashboard, click **Add charts**.

4. Click right angle bracket to expand the **Delivery Flow** chart list.
5. Click **Add +** for **Cycle Time KPI** to add the chart to the dashboard.
6. The **Cycle Time KPI** chart is displayed with the following default status rule:

Table 1176. Cycle Time KPI

Status rule range	Color
0-5	Green
5-10	Yellow
10-30	Red

Note: The default status rule can be customized as required. See custom status rule for cycle time KPI on page 3173.

To set custom status rule for cycle time KPI, complete the following steps:

1. On the **Home** page, click **Settings** on upper-right corner of the page.
2. Click **Insights > KPIs**.
3. To set the status rule for Cycle Time KPI, click an overflow menu and select **Edit**. The **Edit KPI Definition - Cycle Time KPI.** window is displayed.
4. In **Edit KPI Definition - Cycle Time KPI.** window, click **Add status rule** to add a status rule.
5. In **Rule name** field, enter the name for the rule.
6. In **Rule** section in **Edit KPI Definition - Cycle Time KPI.**, select the range and color for the rule status.
7. Click **Add row** to add range for custom status rules. You can select the color for each range of custom status rule.

Note: You can add, edit, and delete status rules.

8. In **Scope > Teams** field, select the teams to apply the status rule.
9. In **Scope > Value streams** field, select the value stream to apply the status rule.
10. In **Scope > Applications** field, select the application to apply the status rule.
11. Click **Confirm**.
12. Click **Submit**.

Lead Time KPI

To add **Lead Time KPI** chart to dashboard, complete the following steps:

1. On the navigation bar, click **Insights**.
2. Select the required dashboard.

3. Under the dashboard, click **Add charts**.
4. Click right angle bracket to expand the **Delivery Flow** chart list.
5. Click **Add +** for **Lead Time KPI** to add the chart to dashboard.
6. The **Lead Time KPI** chart is displayed with the following default status rule:

Table 1177. Lead Time KPI

Status rule range	Color
0-5	Green
5-10	Yellow
10-30	Red

Note: The default status rule can be customized as required. See custom status rule for cycle time KPI on page 3174.

To set the status rule for lead time KPI, complete the following steps:

1. On the **Home** page, click **Settings** on upper-right corner of the page.
2. Click **Insights > KPIs**.
3. To set the status rule for Lead Time KPI, click an overflow menu and select **Edit**. The **Edit KPI Definition - Lead Time KPI.** window is displayed.
4. In **Edit KPI Definition - Lead Time KPI.** window, click **Add status rule** to add a status rule.
5. In **Rule name** field, enter the name for the rule.
6. In **Rule** section in **Edit KPI Definition - Lead Time KPI.**, select the range and colour for the rule status.
7. Click **Add row** to add range for custom status rules. You can select the color for each range of custom status rule.

Note: You can add, edit, and delete status rule.

8. In **Scope > Teams** field, select the teams to apply the status rule.
9. In **Scope > Value streams** field, select the value stream to apply the status rule.
10. In **Scope > Applications** field, select the application to apply the status rule.
11. Click **Confirm**.
12. Click **Submit**.

Quality charts

Table 1178. Quality-type charts

Chart	Description
Coverage By Branch	Percent of branches with associated tests executed during testing.
Coverage By Function	Percent of program functions executed during testing.
Coverage By Line	Percent of lines of code executed during testing.
Functional Tests	Number of functional tests by status in a specified period of time.
Static Code Analysis	Number of issues by severity in a specified period of time.
Unit Tests	Number of tests by status in a specified period of time.
API Tests	Number of tests by status in a specified period of time.

Risk charts

Table 1179. Risk-type charts

Chart	Description
Container Vulnerabilities	Number of container vulnerabilities by severity in a specified period of time. The metric can be set to process names, environment, result, type, application, tag, or team. The default metric is result. The data can scoped by data set, process, or version.
Application Vulnerabilities	Number of application vulnerabilities by severity in a specified period of time.

Lead time, cycle time and dev-cycle time table metrics

Lead time, cycle time and dev-cycle time table metrics provide refined granularity for issues.

Lead time, cycle time and dev-cycle time

To effectively manage and improve the delivery process, it is critical to understand the metrics that reflect the different phases of the workflow: Lead time, Cycle time, and Dev cycle time. These metrics provide deeper insights at the issue level for pinpointing to identify the delays and measure efficiency. Each metric provides a different perspective, helping both technical and business teams align performance and delivery expectations.

Name	Description
Lead Time	The total time from initial request or planning of the work to its completion. This includes all stages—waiting, development, testing, and delivery.
Cycle Time	The time from the start of actual work to the delivery of the final output to the customer. It focuses only on the active execution phase.
Dev Cycle Time	The time from the start of the development team work on a task to the end of the development phase, just before it is released to production.

Viewing the Lead time, Cycle time and Dev-cycle time metrics

You can view the Lead time, Cycle time and Dev-cycle time in the **Insights** page on Measure dashboard.

1. Click **Insights > All Dashboards**.
2. Click to display the dashboard card layout view. Alternatively, you can click to display the dashboard list layout view.
3. Click on the name of required dashboard on the **Insights** page.
4. Click

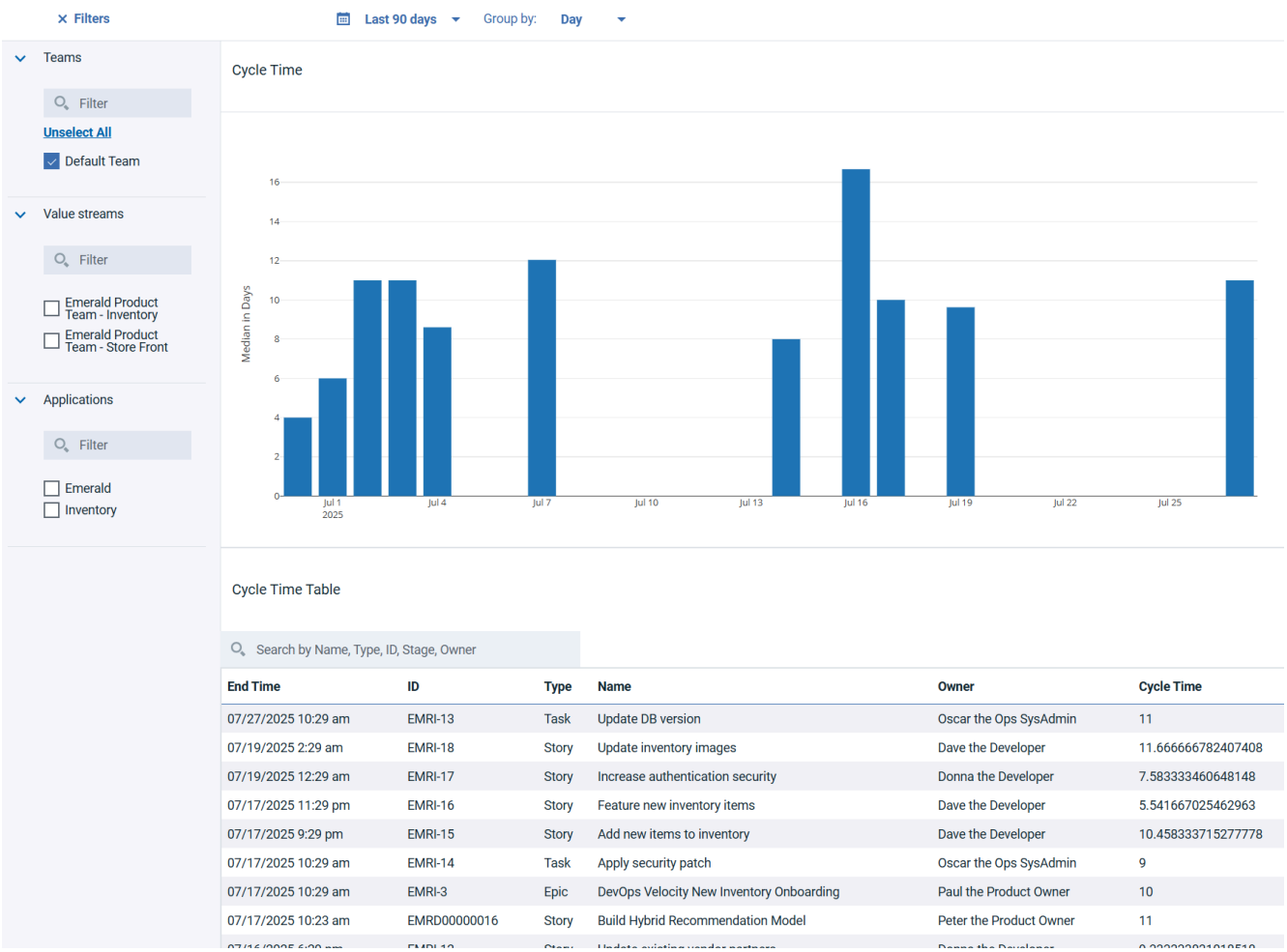
, and expand **Delivery Flow** option.
5. Click

for **Lead Time**, **Cycle Time**, and **Dev Cycle Time**.
6. Click **Save Dashboard**.

   The **Lead Time**, **Cycle Time**, and **Dev Cycle Time** are added to the dashboard. You can view the metric table with graphical representation of the **Lead Time**, **Cycle Time**, and **Dev Cycle Time**

Analyzing the Lead Time, Cycle Time, and Dev Cycle Time in the dashboard

The following example shows the Cycle time in the graphical presentation for the selected Teams, Value streams, and Applications.



The **Lead Time Table**, **Cycle Time Table**, and **Dev Cycle Time Table** display the fields detailed in the following table:

Field	Description
End Time	Date and time completed
Id	Issue number
Type	Issue type including task, story, bug, and so forth
Name	Issue descriptive name
Owner	Owner name
Lead Time	Lead time metric in days. Note: The lead time only appears in the **Lead Time Table**.
Cycle Time	Cycle time metric in days. Note: The cycle time only appears in the **Cycle Time Table**.

Field	Description
Dev Cycle Time	Cycle time metric in days. Note: The dev-cycle time only appears in the **Dev Cycle Time Table**.

Agile Churn metric

The Agile Churn metric is the ratio of the sum of the values of story points added and removed to the value of the planned story points of a sprint. It calculates the percentage of the change in planned scope for any sprint. By using this metric, you can learn how the change in planned scope impacts the progress and completion of the sprint.

Based on the Agile Churn metric data of any sprint, you can plan or organize the planned scope for any future sprints. You can learn the issue details of the story points that are added and removed to the active sprint in the metrics data table.

Note: Currently, Measure supports Agile Churn metric calculation only for the Jira integration. You must integrate or upgrade your Jira integration with plugin 2.3.24 or later for the accuracy of Agile Churn metric calculation.

Measure calculates the Agile Churn metric by using the following formula:

$$\text{Agile Churn} = \frac{\text{Story points added} + \text{Story points removed}}{\text{Planned story points}} \times 100$$

Where:

- **Story points added:** The total value of the story points of the issues that are added to the sprint.
- **Story points removed:** The total value of the story points of the issues that are removed from the sprint.
- **Planned story points:** The total value of the story points of the issues of the sprint that are present at 24 hours of offset time after sprint start time.

Note: Agile Churn metric is calculated after the completion of 24 hours of offset time from the sprint start time. Any story points that are added or removed within 24 hours of offset time is not be considered for Agile Churn metric calculation.

Here is an example of the calculation:

Consider a sprint had a total planned story points of 50 at the end of 24 hours of offset time completion after sprint start time.

An issue with story point of 5 is added to the sprint after completion of 24 hours of offset time from the sprint start time.

An issue with story point of 7 is removed from the sprint after the completion of 24 hours of offset time from the sprint start time.

Therefore, Measure calculates the Agile Churn metric as follows:

$$\text{Agile churn} = \frac{5+7}{50} \times 100 = \frac{12}{50} \times 100 = 24\%$$

Display of the Agile Churn metric in Measure

In Measure, the Agile Churn metric is displayed on the Dashboards and Value Streams pages:

- **Dashboards page:** Agile Churn is displayed in the Dashboards by using a graph, which dynamically shows the average churn percentage over time. You can group by days, month, week, quarter, year or with custom dates.
- **Value Streams page:** The **Value Streams** page shows the Agile Churn for changes in a specific value stream. It shows the percentage of churns in the value stream over the last 90 days.

  The result is presented on the Value Streams page, which helps stakeholders to view the overall stability and reliability of the deployment process for the value stream.

Adding the Agile Churn metric chart to Measure Dashboard

Perform the following steps to add the Agile Churn metric chart to the dashboard:

1. Click **Insights > All Dashboards**.
2. Perform any of the following steps:
   - Click any existing dashboard to open the dashboard.
   - Click **Add new dashboard**, provide the name and description for the dashboard, and then click **Add**.

   The dashboard is displayed.
3. Click **Add charts** and in the **Delivery Flow** category, click **Add** for the **Agile Churn** metric.

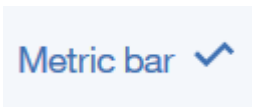
   The chart is displayed on the dashboard.

   If the dashboard already contains charts, the new one is inserted in the top row. Data that matches the filter condition populates the chart.

Adding the Agile Churn metric in the value stream

Perform the following steps to add Agile Churn metric in the value stream:

1. On the Home page, click **Value streams**.

   The **Value Streams** page is displayed.
2. Click 

to display the metric bar.
3. Click to display the **Add Metrics** window.

4. In the **Delivery Flow** category, click **Add** for the **Agile Churn** metric.

   The Agile Churn metric is displayed in the value stream.

Change Failure Rate metric

The Change Failure Rate (CFR) is the ratio of the number of failed changes to the total number of changes deployed within a specified time. It is expressed as a percentage, which helps teams to understand the rate at which their deployments are successful or failed.

The CFR is a key metric used to understand how the changes introduced in software are reliable. It calculates the percentage of changes that fail, which requires rollbacks, hotfixes, or other actions. As part of the DORA metrics, CFR measures the percentage of changes that need to be rolled back, fixed with a hotfix, or require other actions. This metric helps organizations to evaluate the stability of their deployment process.

Note: Currently, Measure supports CFR metric calculation only for the ServiceNow integration. You must integrate or upgrade your ServiceNow integration with plugin version 1.1.13 or later with custom field mapping for CFR metric calculation.

Measure calculates the CFR metric by using the following formula:

$$\text{Change Failure Rate (CFR)} = \frac{\text{Number of failed changes}}{\text{Total number of changes}} \times 100$$

Where:

- **Number of failed changes:** The total number of failed changes such as rollbacks or incidents triggered by those changes.
- **Total number of changes:** The total number of changes made during the given time, such as code deployments, updates, rollbacks, incidents, or releases.

Here is an example of the calculation

A project had a total of 50 changes deployed, out of which 5 resulted in failures.

Therefore, Velocity calculates CFR as follows:

$$\frac{5}{50} \times 100 = 10\%$$

Display of the CFR metric in Measure

In Measure, the CFR metric is displayed on the Dashboards and Value Streams pages:

- **Dashboards page:** CFR is displayed in the Dashboards by using a graph, which dynamically shows changes in the failure rate over time. You can group by days, month, week, quarter, year or with custom dates.
- **Value Streams page:** The **Value Streams** page shows the CFR for changes in a specific value stream. It shows the percentage of failed changes in the value stream over the last 90 days.

  The result is presented on the Value Streams page, which helps stakeholders to view the overall stability and reliability of the deployment process for the value stream.

Adding the CFR metric chart to Measure Dashboard

Perform the following steps to add the CFR metric chart to Dashboards:

1. Click **Insights > All Dashboards**.
2. Perform any of the following steps:
   - Click any existing dashboard to open the dashboard.
   - Click **Add new dashboard**, provide the name and description for the dashboard, and then click **Add**.

   The dashboard is displayed.
3. Click **Add charts** and in the DORA category, click **Add** for the CFR metric.

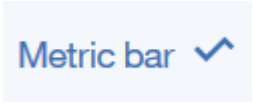
   The chart is displayed on the dashboard.

   If the dashboard already contains charts, the new one is inserted in the top row. Data that matches the filter condition populates the chart.

Adding the CFR metric in the value stream

Perform the following steps to add CFR metric in the value stream:

1. On the Home page, click **Value streams**.

   The **Value Streams** page is displayed.
2. Click 

to display the metric bar.
3. Click to display the **Add Metrics** window.
4. In the DORA category, click **Add** for the CFR metric.

   The CFR metric is displayed in the value stream.

Flow Efficiency metric

The Flow Efficiency metric is the ratio of actual development time to cycle time. It is expressed as a percentage, which helps teams to understand their performance. A higher flow efficiency indicates a smoother process with fewer delays, while lower values indicate inefficiencies that might require attention to identify and eliminate bottlenecks in the workflow.

The Flow Efficiency metric is a key metric used to assess the reliability and success of the changes deployed within a system. This metric provides insight into the efficiency of work in the development pipeline. By using the Flow Efficiency metric, teams can create a more predictable and effective delivery process. Therefore, you can get better-quality results and increased delivery efficiency.

Measure calculates the Flow Efficiency metric by using the following formula:

$$\text{Flow Efficiency} = \frac{\text{Story points x 7.5}}{\text{Cycle time}} \text{ x 100}$$

Where:

Story points: The total value of the story points of the issues.

Cycle time: The total time taken for each issue to progress from start to completion.

Here is an example of the calculation:

A work item had 6 story points, and the cycle time was 18.5.

Therefore, the Flow Efficiency metric is calculated as follows:

$$\frac{6 \text{ x } 7.5}{18.5} \text{x } 100 = 243.24\%$$

Display of the Flow Efficiency metric in Measure

In Measure, the Flow Efficiency metric is displayed on the **Dashboards** and **Value Streams** pages:

- **Dashboards** page: The Flow Efficiency metric is displayed in the **Dashboards** by using a graph, which dynamically shows flow efficiency data in percentage over time. You can group by days, month, week, quarter, year, or with custom dates.
- **Value Streams** page: The **Value Streams** page shows the Flow Efficiency metric in a specific value stream. It shows the ratio of actual development time over the cycle time in percentage by taking the number of overall work items in the value stream in the last 30 days.

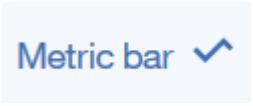
  The result is presented on the **Value Streams** page, which enables stakeholders to view the overall stability and reliability of the deployment process for that specific value stream.

Adding the Flow Efficiency metric in the value stream

Perform the following steps to add the Flow Efficiency metric for a value stream:

1. On the Home page, click **Value streams**.

   The value stream page is displayed.

2. On the **Value Streams** page, click 

to display the metric bar.
3. Click to display the **Add Metrics** window.
4. In the **Delivery Flow** category, click for the **Flow Efficiency** metric.

   The Flow Efficiency metric is displayed in the **Metrics** bar on the value stream page.

Adding the Flow Efficiency metric chart to the Dashboard

Perform the following steps to add the Flow Efficiency metric chart on **Dashboards**:

1. Click **Insights > All Dashboards**.
2. Perform any of the following steps:
   - Click any existing dashboard to open the dashboard.
   - Click **Add new dashboard**, provide the name and description for the dashboard, and then click **Add**.

   The Dashboard is opened.

3. Click **Add charts** and in the **Delivery Flow** category, click **Add** for the **Flow Efficiency** metric.

   The chart is displayed on the dashboard.

GenAI metrics - Tech Preview

Overview

Disclaimer:

This release contains access to the GenAI metrics feature in Measure as a Tech Preview. The Tech Preview is intended for you to view the capabilities of GenAI metrics offered by Measure, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL® provides the information without obligation of support and "as is" without warranty of any kind.

The GenAI metrics provides you the insights about GitHub Copilot usage within an organization. This feature retrieves user activity, engagement levels, language-specific completions, and other Copilot adoption metrics directly from the GitHub API and syncs them into Measure

The Gen AI metrics helps organizations track how developers are using AI-assisted coding and measure productivity impact through structured metrics.

The following GenAI metrics can be added and displayed within the Measure insights section.

- **GenAI Line Acceptance**
- **GenAI User Adoption**
- **GenAI Chat Events**
- **GenAI Code Acceptance**
- **GenAI PR Events**

Prerequisites

You must integrate or upgrade your Github integration with plugin version 1.5.4 or later with Copilot metrics collection enabled during integration. For more information, refer to Github plugin documentation.

Display of the GenAI metrics in Measure

In Measure, the GenAI metrics are displayed on the **Dashboards** page under **GenAI** category:

Adding the GenAI metric chart to the Dashboard

Perform the following steps to add the GenAI metric charts on **Dashboards**:

1. Click **Insights > All Dashboards**.
2. Perform any of the following steps:
    - Click any existing dashboard to open the dashboard.
    - Click **Add new dashboard**, provide the name and description for the dashboard, and then click **Add**.

   The Dashboard is opened.
3. Click **Add charts** and in the **GenAI** category, click **Add** for the available GenAI metrics collected from GitHub Copilot.

   The chart is displayed on the dashboard.

Insights reports

Insights reports are generated for the data that is aggregated in the **Value streams**, **Pipelines**, and **Release** categories. You can generate reports by clicking a button and then shared with the required stakeholders. You can run the following insights reports in Measure.

Running insights reports

The insights reports generate reports for the data aggregated into Measure.

To run an insights report, perform the following steps:

1. Click **Insights > Reports**.
2. Click **Add report** to add the required report from the report templates.
3. After adding the report, click **Run Report** to open the **RUN REPORT** window.
4. Select the required property for running the report.

   Note: You can set the property as default by clicking **Set as Default**.

5. Click **Confirm** to run the report.

The insights report page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.
- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the insights report page is displayed.

Downloading sample report

You can download the sample insights report by performing the following steps:

1. Click **Insights > Reports**.
2. Click **Download sample report** to download sample report.

The downloaded sample report folder contains `report.json` file with sample properties to generate custom reports. You must read the instructions available in README file for generating custom reports.

AppScan Enterprise scan report

The AppScan Enterprise (ASE) scan report displays a summary of vulnerabilities identified in the scans conducted by an AppScan Enterprise instance for which an Measure integration has been configured.

The AppScan Enterprise report provides the information about the quality and severity of security issues. The report displays common issue types, issue type by application, and scans by application for the selected duration of time.

To run the **AppScan Enterprise scan** report, perform the following steps:

1. Click **Insights > Reports**.
2. Select the **AppScan Enterprise scan report** field under the **Report Templates** heading on the left side of the page.

Note: When the **AppScan Enterprise scan report** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click **Run Report** to open the **RUN REPORT** window.
4. Enter the following information:
    - **Start date** - Select the start date for the scan report
    - **End Date** - Select the end date for the scan report
5. Click **Confirm** to run the report.

The **AppScan Enterprise scan report** page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.
- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the **ASE scan report** page is displayed.

Release readiness report

The **Release readiness** report details criteria based on internal standards and compliance requirements.

The **Release readiness** report page provides comprehensive details for assessing risk of a specific version prior to its release into the production environment. In order to evaluate release readiness for a new version candidate, the report assists in evaluating the following areas:

- Are release preparations final and is the version meeting all quality criteria?
- Did the reviewers submit the required approvals?
- Is the version thoroughly and properly tested against the quality criteria?
- Do specific issues exist for this version now that require additional attention?

To run the **Release readiness** report, perform the following steps:

1. Click **Insights**, and then select **Reports**.
2. Select the **Release readiness** field under the **Templates** heading on the left side of the page.

Note: When the **Release readiness** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click the **Run report** button to display the **RUN REPORT** window.
4. In the **Pipeline / value stream** drop-down, select the required value stream and pipeline from the list.
5. In the **Stage containing release candidates (optional)** field, enter the pipeline environment containing the release candidates.
6. In the **Targeted release stage (optional)** field, enter the environment.

Note: To set the report inputs as default, click the **Set as Default** button.

7. Click the **Confirm** button to run the report.
8. If the report ran successfully, in the **Status** field of the table below the **Run report** button, a green circle and **Complete** is displayed.

Note: If a red circle and **Failed** is displayed, run the report again.

9. After the report runs successfully, in the line item for the report in the table, click **Results** to view the **Release readiness** report.

A sample **Release Readiness Report** page is shown below.

Release readiness report

Aug 7th 2024, 2:06:24 pm UTC

Value stream: Ai

This report collects and analyzes data from Ai value stream and related delivery pipeline including Insights quality data to assess the current release readiness for : Ai | DEV release.

Release: Ai | DEV release

Scheduled date: Aug 7th, 2024

Application: Jenkins

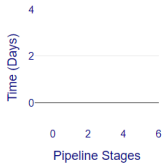
CANDIDATE	CURRENT VERSION
QA	PROD
#41	#40
Aug 7th, 2024	Aug 7th, 2024

Readiness score

Computed readiness score

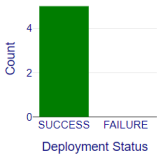
Time in stage

Time version candidate spent in each stage



Deployment success rate

Last 30 days success rate: 100%



Release summary

Release summary

Summary of the release

✓ **Improved User Experience:** The release focused on enhancing the user interface and experience by redesigning interactive elements and refreshing the global design. This included modifications to the web client interface for better usability (VEL-247).

✓ **Performance Optimization:** Several tasks were completed to optimize database queries and implement caching mechanisms to improve system performance and scalability (issues_id).

✓ **Bug Fixes and Quality Assurance:** The release included a thorough review of documentation and addressed issues related to the user interface, image display problems, and container start-up errors (issues_id).

✓ **New Feature Development:** The release also included the development of new features and functionality to enhance the overall user experience, such as integrating with third-party services and adding support for mobile devices (issues_id).

Warning: 2 out of 2 issues have no description. The summary may not be accurate.

Release content

Candidate version

Candidate version currently in QA

Id: 66b34c285a1f1f4accd0207f
Name: #41
Source: ucv-ext-jenkins
Created on: Aug 7th, 2024

Name	Status
#41	Success

Issues:

Id	Type	Issue	Sprint	Owner
VEL-248	Bug	Pavan05	Sprint28	
VEL-247	Story	pavan04	Sprint27 Sprint28	

Pull requests:

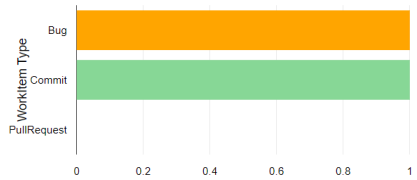
Id	Created	Pull Request	Owner
2008128053	Aug 7th, 2024	VEL-248 Update kirti.txt	pujarpavan
2008097599	Aug 7th, 2024	VEL-247 Update kirti.txt	pujarpavan

Commits:

Name	Owner
Merge pull request #61 from pujarpavan/pujarpavan-patch-44 VEL-248 Update kirti.txt	Pujarpavan
Update kirti.txt	Pujarpavan
Merge pull request #60 from pujarpavan/pujarpavan-patch-43 VEL-247 Update kirti.txt	Pujarpavan
VEL-247 Update kirti.txt	Pujarpavan

Changes introduced since version (#40)

Delta between current inventory in target environment and this candidate



Issues:

Id	Type	Issue	Sprint	Owner
VEL-248	Bug	Pavan05	Sprint28	

Pull Requests:

Id	Created	Pull Request	Owner

Commits:

Date	Name	Owner
Aug 7th, 2024	Update kirti.txt	Pujarpavan

Individual Measures:

Owner	Developement Item Completed (Total)	Completed Story Points (Total)	Pull Request Count	Commit Count	Avg. Dev Cycle Time (Days)	Avg Dev Cycle Time Per Story Point (Days)
Pujarpavan	1	2	0	1	4.44	2.22

Contributors (1)

Owners of introduced changes

Pujarpavan

Release quality & security

Current version quality & security metrics

Insights quality metrics associated with candidate

Performance error rate
Test version 8830

numerator	denominator	percent
0	30015	0

Performance response time
Test version 8830

Mean
3.67

Code coverage by line
Test version 8830

numerator	denominator	percent
63	100	63

IMAGE_SCANS
Test version 8830

Low	Medium	High	Critical
61	57	21	1

Functional Tests
Test version 8830

Failed	Skipped	Passed	ignored
0	0	15	0

Security Static Scan
Daily Scan

High	Medium	Low	Info
2	4	1	10

Unit Tests
Test version 8830

Passed	Failed
6	0

Static code analysis
Daily Scan

High	Medium	Low	Info
2	7	1	0

Current application quality & security metrics

Insights quality metrics associated with HCL ARP & aliases

Code coverage by line
ARP
Test version 8830
Oct 20th, 2021

numerator	denominator	percent
63	100	63

Static code analysis
ARP
Daily Scan
Oct 20th, 2021

High	Medium	Low	Info
2	7	1	0

Security Static Scan
HCL ARP
Daily Scan
Oct 20th, 2021

High	Medium	Low	Info
2	4	1	10

Functional Tests
HCL ARP
Test version 8830
Oct 20th, 2021

Failed	Skipped	Passed	ignored
0	0	15	0

Pipeline gates

Stage gate statuses

PROD

Name: RM Approval
Type: Approval
Approver: admin@admin.com

Name: Functional Test
Type: Automated

QA

Name: QA Approval
Type: Approval
Approver: admin@admin.com

Name: Unit Test
Type: Automated

Release process

The main sections of the Release readiness report include the following:

- **Report header**
- **Release summary**
- **Release content**
- **Release quality & security**
- **Release process**

Subdivisions of the main sections listed above are detailed in the table below.

Table 1180. Report sections details

Main section	Subdivision
Report header	**Value stream** name, **Release** name with **Scheduled date**, **Application** name with **CANDIDATE** and **CURRENT VERSION** details, **Readiness score**, **Time in stage**, **Deployment success rate**
Release summary	**Release summary**
Release content	**Candidate version**, **Changes introduced since version (number)**, **Contributors**, and **Individual Measures**.
Release quality & security	**Approvals**, **Current version quality & security metrics**, **Current application quality & security metrics**, **Pipeline gates**
Release process	**Release activites**, **Value stream - highest priority items opened**

Report header

Under the **Application** name in the report header, the **CANDIDATE** and **CURRENT VERSION** details including environment, version, and date are displayed for each. These two versions are compared with one another with the results listed in the report sections contained in the table above. The **Readiness score**, **Time in stage**, and **Deployment success rate** details are also displayed in the report header.

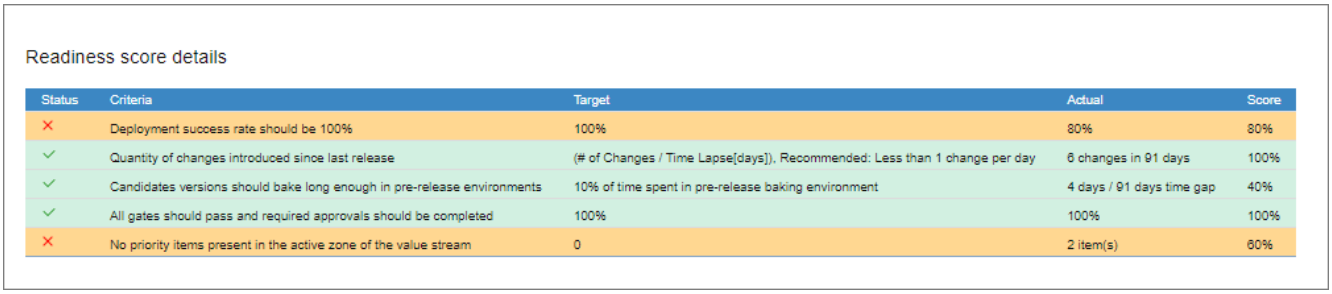
Readiness score

The **Readiness score** view provides a detailed readiness score breakdown for the candidate version and is viewed by hovering over the icon in the bottom right hand corner of the section. The **Readiness score details** are listed in tabular form with the fields detailed and shown in the following table and figure respectively.

Table 1181. Readiness score details

Field	Description
Status	Indicates the state of the specific criteria as success or fail with the green ⎫ or red ⎭ symbols respectively.
Criteria	Specific area of focus for this release candidate.
Target	Based on the criteria, what is the optimal value for success?
Actual	Based on the criteria, what actual value was achieved?
Score	The percentage of success between the **Score** and **Actual** values.

Figure 28. Readiness score details



Release summary

In the **Release summary** section, you can view the composite summary of the works items associated with the release. This section provides insights on the features, enhancements, defects, and the overall status of release activities.

Note: You must install the GenAI Summary Release plug-in to view the **Release summary**.

Release content

The **Candidate version** section identifies details for the specific build including names, issues, PR and commits. The **Changes introduced since version (number)** section lists the items that are added to the **CURRENT VERSION** to represent the **CANDIDATE** version including issues, PR and commits. The names and number of owners providing changes for the aforementioned items in the new version are listed in the **Contributors** section. Links are provided in each of the aforementioned sections to view information for specific items. You can also see the individual measures for the contributors of the release. See the **Individual Measures** on page 3190 section for more details.

Individual Measures

The **Individual Metrics** section, available in the **Release Readiness Report**, provides detailed insights into the performance and involvement of each contributor in the release. This data is essential for engineering managers, product owners, and team leads to assess individual efficiency, promote accountability, and drive continuous improvement.

For every contributor who has participated in the release, the following metrics are displayed:

- **Number of Issues Completed**

  Tracks the number of tasks (stories, bugs, or technical items) a contributor has completed. This metric helps evaluate task throughput and individual contribution levels.

- **Total Story Points Completed**

  Measures the total effort delivered by a contributor based on the story points assigned to completed work items. This metric supports capacity planning and performance evaluation.

- **Number of Pull Requests Created**

  Indicates the frequency at which a contributor submits the code changes for review. This metric provides insight into coding activity and collaboration with peers.

- **Total Number of Commits**

  Shows the number of code commits made by a contributor, reflecting consistency, development cadence, and active involvement in coding tasks.

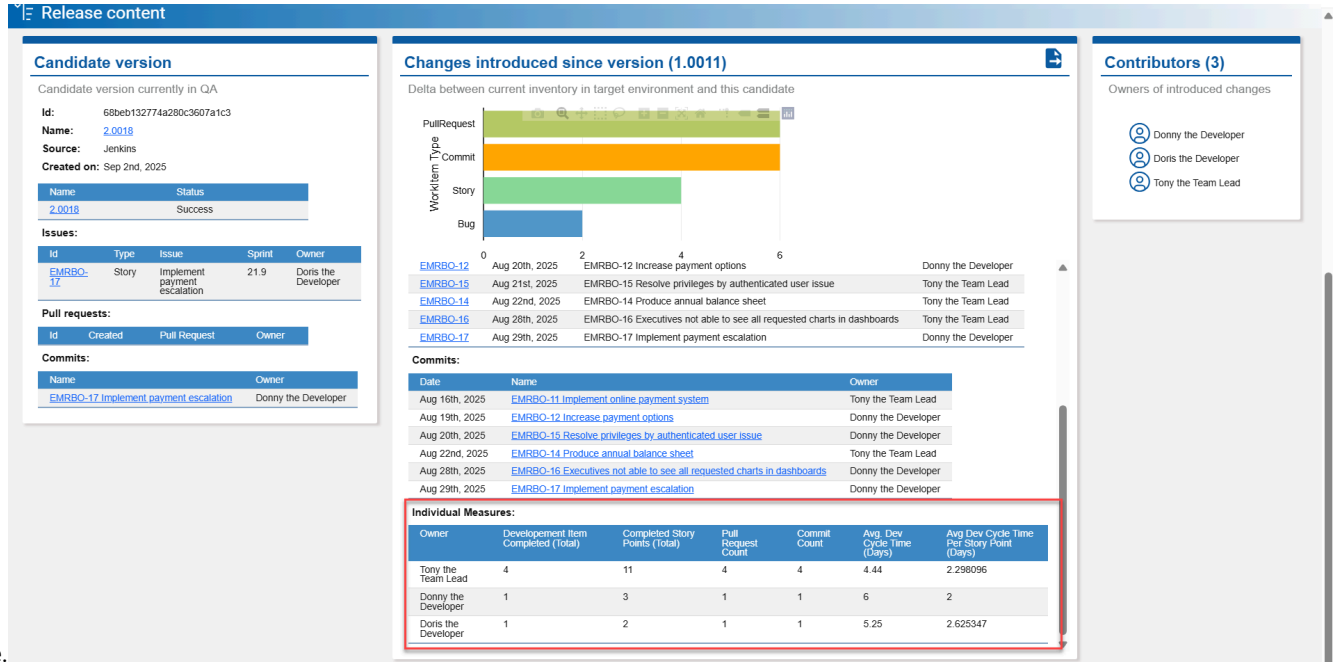
- **Dev Cycle Time**

  Measures the duration a contributor takes to complete a work item, calculated based on the specific stages configured for Dev Cycle Time within your value stream. This metric is useful for identifying delays, inefficiencies, or areas needing support or process improvement.

- **Dev Cycle Time per Story Point**

  Normalizes Dev Cycle Time by story point size, allowing teams to evaluate individual efficiency relative to task complexity. This metric helps in identifying high-performing contributors or highlighting potential mismatches between estimated and actual effort.

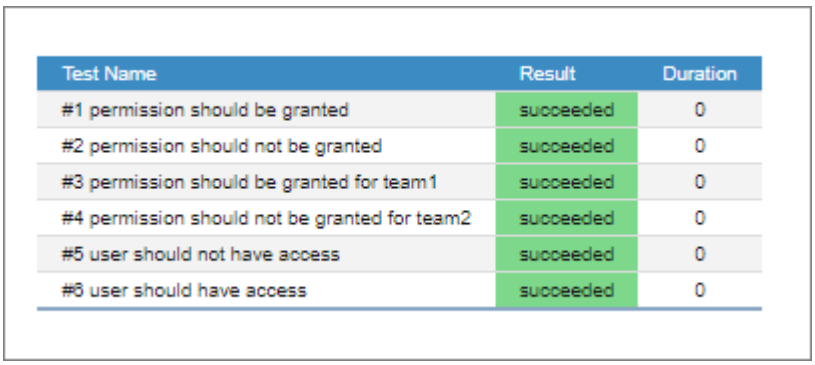
Individual Measures are displayed in a tabular format within the **Release Readiness Report**, as shown in the following example, offering a clear overview of the contributions and performance metrics of each contributor for the release.



Release quality & security

This section focuses on how extensively the candidate version was tested along with the test results. The unique test results for the **CANDIDATE** version are found in the **Current version quality & security metrics** section. Recent Appscans affecting the application are viewed in the **Current application quality & security metrics** section and may not be directly tied to the candidate version. Links to specific metric pages are provided on the right side of the metrics for each of the aforementioned sections. You can also hover over the test results blocks to display a test results table including **Test Name**, **Result**, and **Duration** as shown in the following figure.

Figure 29. Test results table



Note: Only test blocks containing data will display the tabular pop-up shown in the above figure.

In the **Pipeline gates** section, you can view approvals and quality checks across the pipeline phases.

Release activities

In this section, you can view scheduled or releases in progress and any prerelease activities necessary prior to moving into the production environment. Links to specific release activities are provided in this section.

Value stream - highest priority items opened

This section monitors specific issues in the value stream for the candidate version that require attention and resolve prior to release. Links to specific work items are provided in this section.

Reporting on security

The **Security Report** displays roles of all users based on the teams or groups they belongs to.

The **Security Report** provides a comprehensive list of all users with the teams they belong to, the roles they are assigned to, and the groups they associated with. Users can be filtered by categories such as, **By Users** , **By Teams** , **By Roles**, and **By Groups**. With the security report, you can view users based on their roles, teams, and groups.

To run the **Security Report**, perform the following steps:

1. Click **Insights > Reports**.
2. Select the **Security Report** field under the **Report Templates** heading on the left side of the page.

   Note: When the **Security Report** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click **Run Report** to open the **RUN REPORT** window.
4. Enter the security report title in the **Report Title** field.

   Note: To set a report title as default, click **Set as Default**.

5. Click **Confirm** to run the report.

The **Security Report** page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

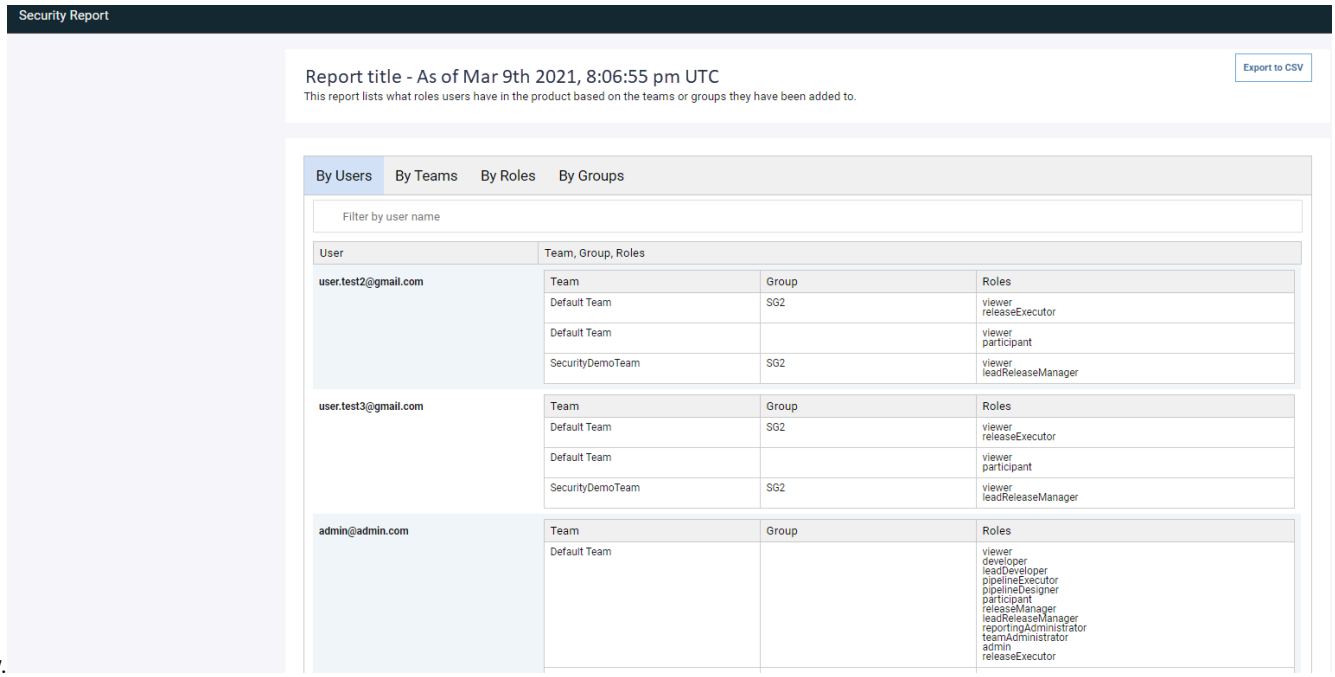
- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.

- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the **Security Report** page is displayed. You must then click **Export to CSV** to export the deployment audit report.

The security report is displayed with four categories:

1. **By Users:** Type the user name in **Filter by user name** to filter users by user name.
2. **By Teams:** Type the team name in **Filter by team name** to filter users by team name.
3. **By Roles:** Type the role name in **Filter by role name** to filter users by role name.
4. **By Groups:** Type the user name in **Filter by group name** to filter users by group name.

A sample **Security Report** page is shown below.



State of the sprint report

The **State of the Sprint Report** page displays all the sprints with related work items for a value stream selected as the output for the report execution.

The **State of the Sprint Report** page provides comprehensive details of all active and inactive sprints for a selected value stream. All of the issues are aggregated by each sprint and grouped according to the stage position of the items in the value stream. Current sprints are identified as **Active** while past and future sprints are identified as **Inactive** on the **State of the Sprint Report** page. For a given sprint in a value stream, data represented in the report includes the following:

- Start, end, and completed dates
- Total issues number
- Percentage issues done
- Story points planned number

- Story points done number
- Issues not pointed number
- Contributors number
- Sprint Burndown chart
- Individual Measures

Additionally, **Pointed vs Not Pointed** and **Planned vs Done (Release)** bar chart graphs for all the sprints in the value stream are displayed at the top of the **State of the Sprint Report** page.

Notes:

- Before you generate **State of the sprint** report, you must install and integrate **Handlebars Reporter** plugin with Measure. For more information, see **Handlebars Reporter** plugin documentation.
- You must use Handlebars Reporter plugin version 2.0.5 or later to get accurate metrics in **Individual Measures** section.

To run the **State of the Sprint** report, perform the following procedure:

1. Click **Insights > Reports**.
2. Select the **State of the Sprint** field under the **Report Templates** heading on the left side of the page.

   Note: When the **State of the Sprint** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click **Run Report** to open the **RUN REPORT** window.
4. Select the required **Value Stream** from the list.

   Note: To set a value stream as default, click **Set as Default**.

5. Based on the **Value Stream** selected, the **Sprints** field selects the latest 10 sprints by default. You can add or remove sprints in the **Sprints** field to generate the report.

   Sprints field value is mandatory to generate the report. You must select minimum one sprint to generate the report.

6. Click **Confirm** to run the report.

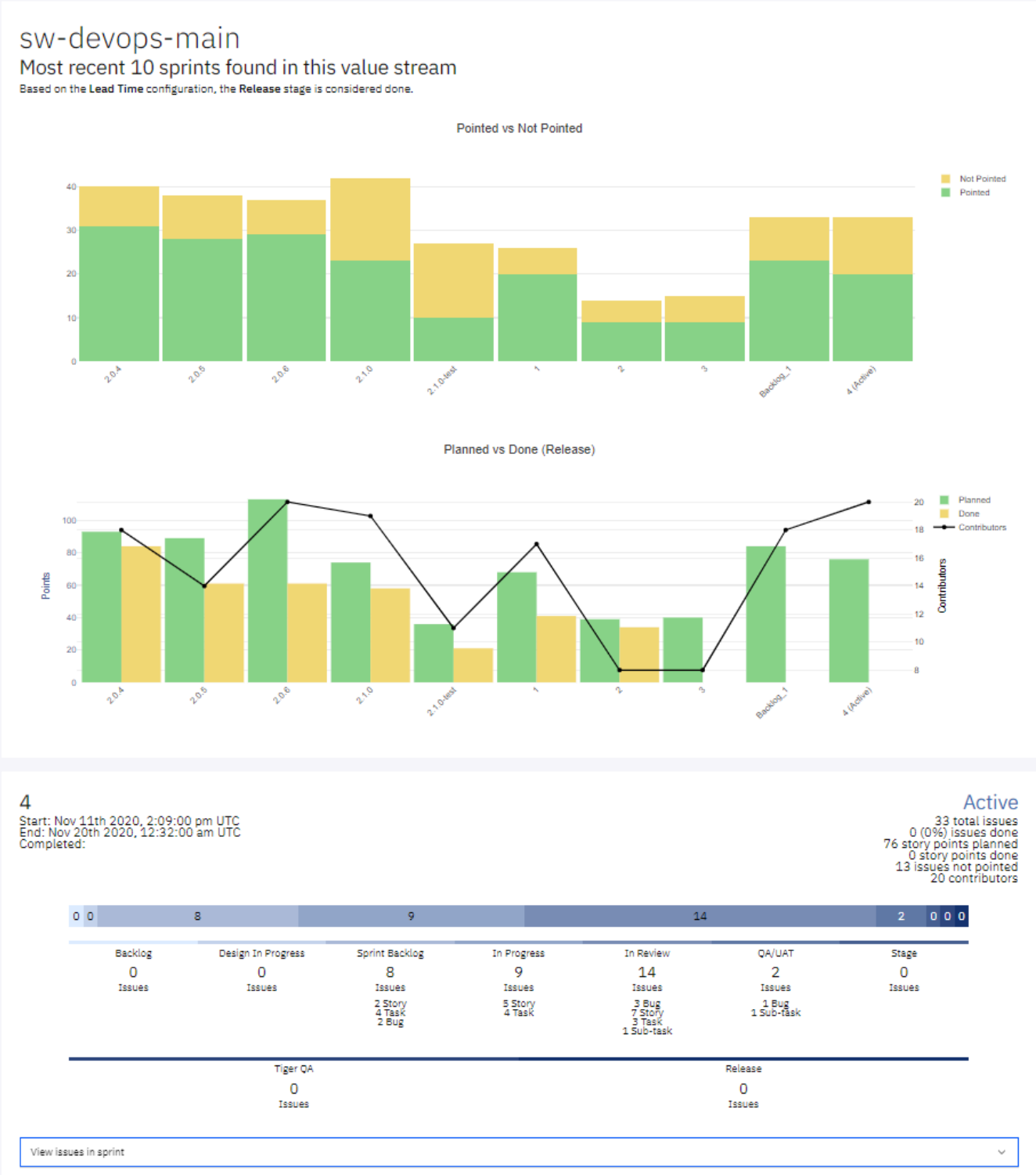
The **State of the Sprint Report** page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

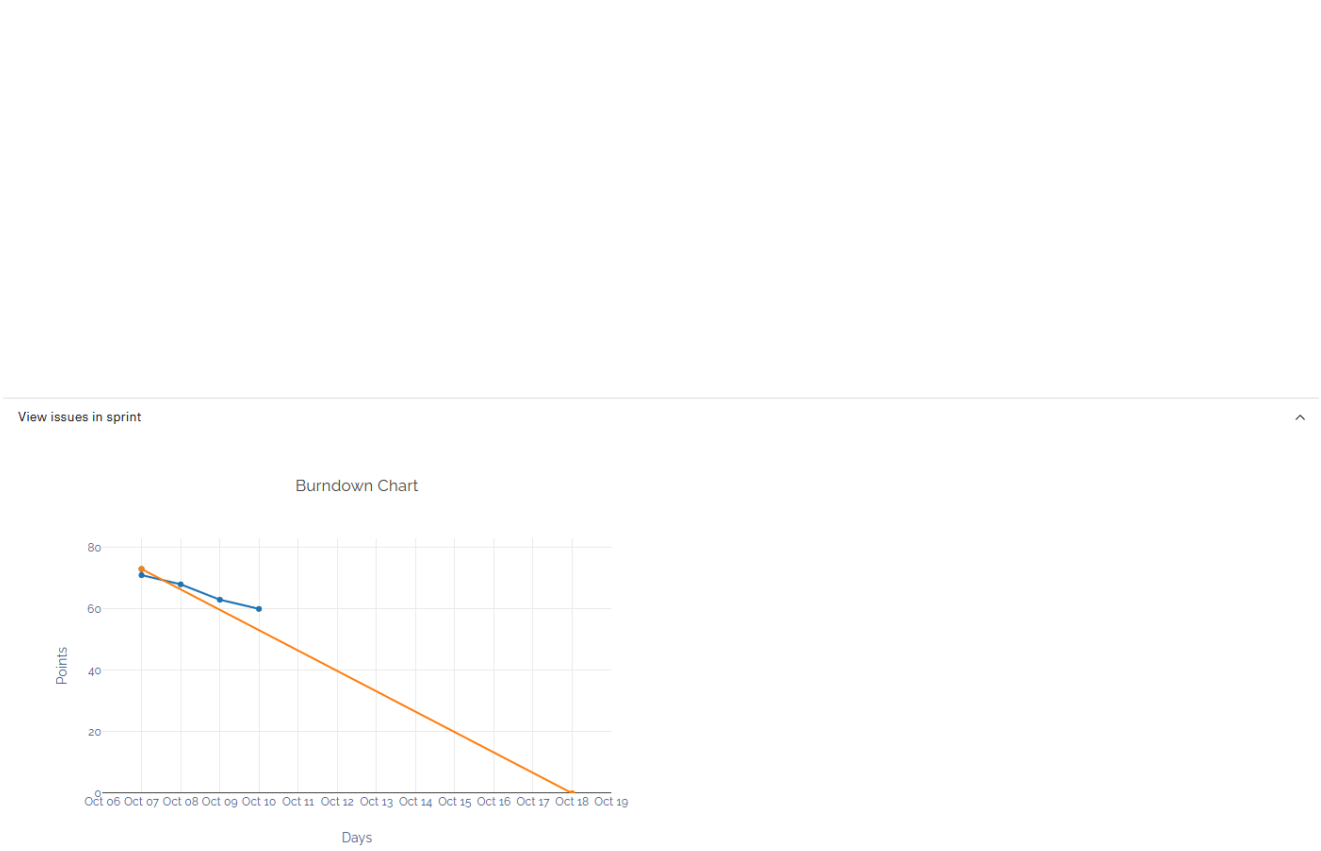
- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.
- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the **State of the Sprint Report** page is displayed.

A sample **State of the Sprint Report** page is shown below.



The **Pointed vs Not Pointed** and **Planned vs Done** bar charts are both interactive and you can hover over any areas of the chart to dynamically display values and plot tool icons providing extended functionality. You can click on different areas of the blue shaded bar chart above to display the stage name with number of related issues.

Click **View issues in sprint** bar to extend and display the burndown chart, related issues for each stage of the value stream and the individual measures for the developers who has contributed to the sprint.



The red vertical bar indicates alerts for issues such as over median time, no owner, and so on. You can click the issue link to open the ticket in a browser window to view detailed information for that item.

Individual Measures

The **Individual Metrics** section, available for each sprint, provides detailed insights into the contributions and performance of each contributor. This data is critical for engineering managers, product owners, and team leads to assess individual efficiency, promote accountability, and guide continuous improvement initiatives.

For every contributor who has participated during a sprint, the following metrics are displayed:

- **Number of Issues Completed**

  Tracks the number of tasks (stories, bugs, or technical items) a contributor has completed. This metric helps evaluate task throughput and individual contribution levels.

- **Total Story Points Completed**

  Measures the total effort delivered by a contributor based on the story points assigned to completed work items. This metric supports capacity planning and performance evaluation.

- **Number of Pull Requests Created**

  Indicates the frequency at which a contributor submits the code changes for review. This metric provides insight into coding activity and collaboration with peers.

- **Total Number of Commits**

  Shows the number of code commits made by a contributor, reflecting consistency, development cadence, and active involvement in coding tasks.

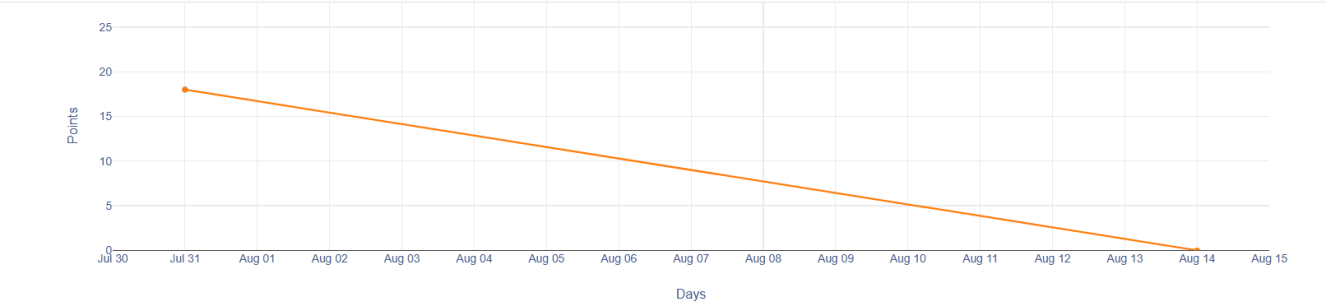
- **Dev Cycle Time**

  Measures the duration a contributor takes to complete a work item, calculated based on the specific stages you've configured for Dev Cycle Time within your value stream. This metric is useful for identifying delays, inefficiencies, or areas requiring support or process refinement.

- **Dev Cycle Time per Story Point**

  Normalizes Dev Cycle Time by story point size, allowing teams to evaluate individual efficiency relative to task complexity. This metric helps in identifying high-performing contributors or flag potential mismatches between estimated and actual effort.

Individual Measures are displayed in a tabular format within the **State of the Sprint** report, as shown in the following example, offering a clear overview of the contributions and performance metrics of each contributor for a given sprint.



PROD				
EMRBO-6 Story - Highest	Done	3 Pts	Enabled PDF conversion and download of reports Business	owner: Doris the Developer creator: Patrick the Product Owner
EMRBO-7 Story - High	Done	5 Pts	Enable account report filter by client ID Business	owner: Tony the Team Lead creator: Patrick the Product Owner
EMRBO-8 Story - High	Done	2 Pts	Produce account payment history screen Business	owner: Tony the Team Lead creator: Patrick the Product Owner
EMRBO-9 Story - Highest	Done	3 Pts	Improve application page rendering performance Internal	owner: Donny the Developer creator: Patrick the Product Owner
EMRBO-10 Story - High	Done	5 Pts	Create executive only dashboards Business	owner: Tony the Team Lead creator: Patrick the Product Owner

Individual Measures

Owner	Developement Item Completed (Total)	Completed Story Points (Total)	Avg. Dev Cycle Time (days)	Avg. Dev Cycle Time Per Story Point (days)	Pull Request Count	Commit Count
Tony the Team Lead	3	12	5.20	1.56	3	3
Doris the Developer	1	3	4.17	1.39	1	1
Donny the Developer	1	3	5.25	1.75	1	1

Deployment audit report

The Deployment audit report displays a summary of the deployments.

The **Deployment audit** report page provides comprehensive details about the deployment frequency, success rate, deployment by application type, environment, and process.

To run the **Deployment audit** report, you must perform the following steps:

1. Click **Insights > Reports**.
2. Select the **Deployment audit** field under the **Report Templates** heading on the left side of the page.

   Note: When the **Deployment audit** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click **Run Report** to open the **RUN REPORT** window.
4. Enter the following information:
   - **Start date** - Start date and time for the report
   - **End date** - End date and time for the report

- **Pipelines** - Select the pipeline
- **And / or pick applications** - Select the application you want to include in report
- **Timezone** - Select the time zone

5. Click **Confirm** to run the report.

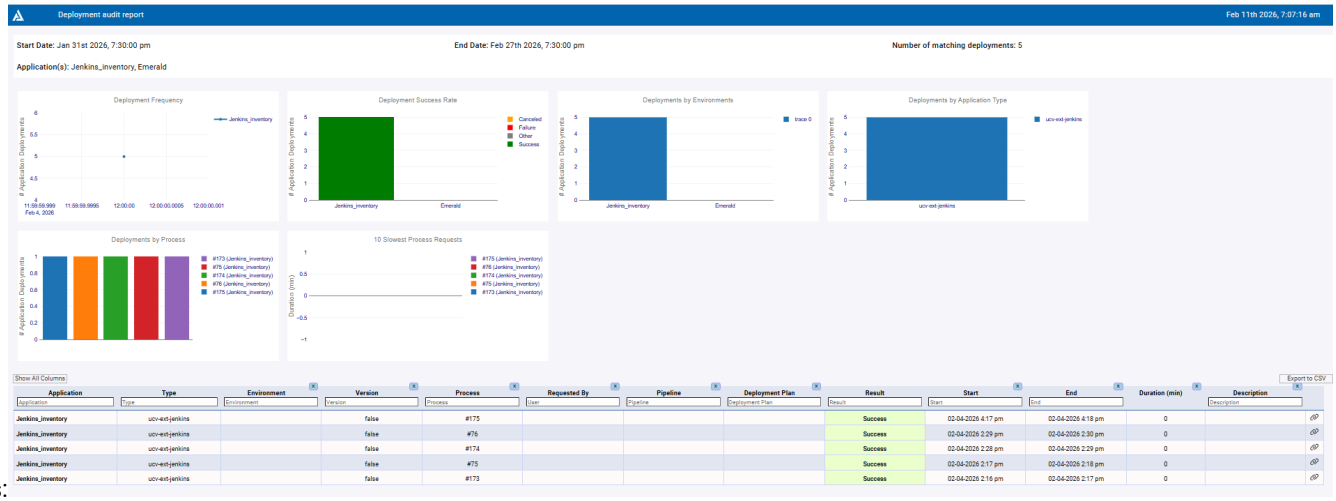
The **Deployment audit report** page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.
- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the **Deployment audit report** page is displayed. You must then click **Export to CSV** to export the deployment audit report.

A sample deployment audit report page is shown as follows:



Pipeline snapshot report

The Pipeline snapshot report displays the snapshot of the pipeline for a specific value stream.

The **Pipeline snapshot** report page provides the details about the pipeline view for a value stream for a particular date.

To run the **Pipeline snapshot** report, you must perform the following steps:

1. Click **Insights > Reports**.
2. Select the **Pipeline snapshot** field under the **Report Templates** heading on the left side of the page.

Note: When the **Pipeline snapshot** field is selected, a vertical blue bar is displayed on the left side of the field to indicate it as the report to run.

3. Click **Run Report** to open the **RUN REPORT** window.
4. Enter the following information:
   - **Value Stream** - Select the value stream from the drop-down list
   - **Snapshot Date** - Select the date for the pipeline snapshot
5. Click **Confirm** to run the report.

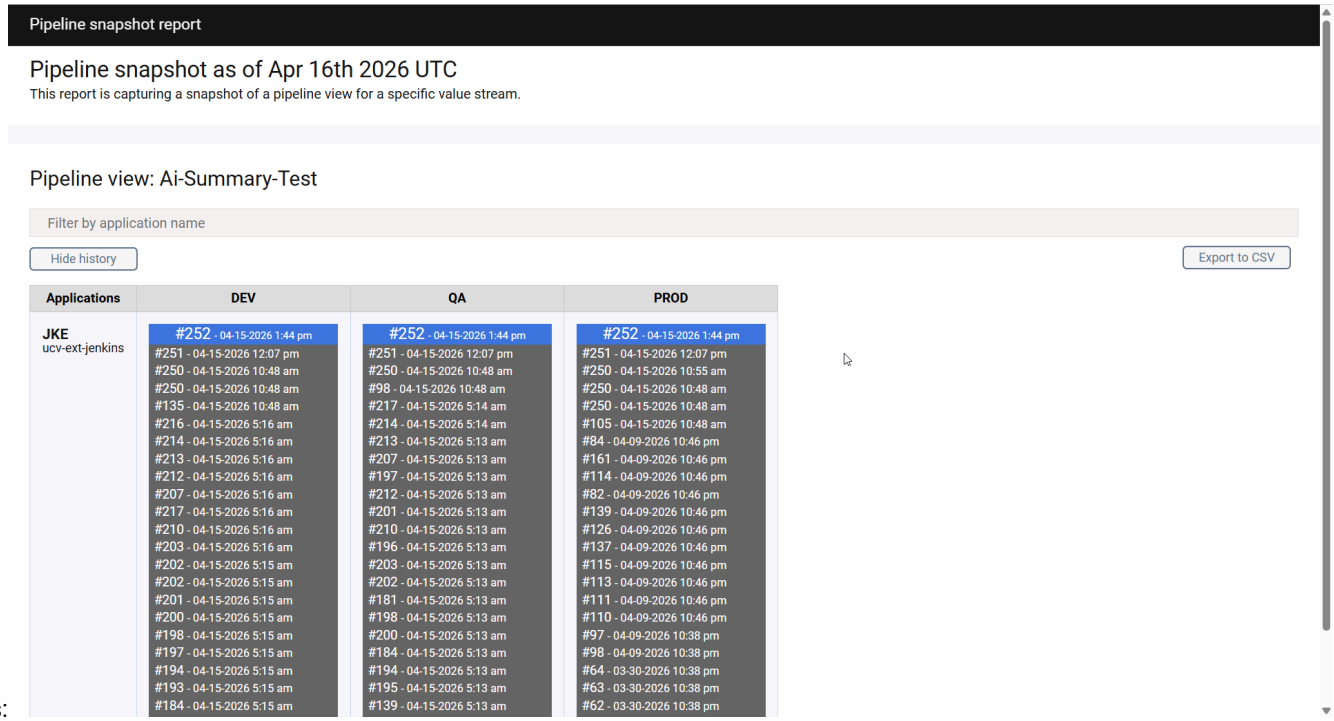
The **Pipeline snapshot** page displays the following information about the report:

- **Date**: Shows the date, time, and email address of the user who ran the report.
- **Status**: Shows the color-coded status: Orange = **STARTED**, green = **COMPLETE**, and red = **FAILED**.

The following elements provide detailed information about the report:

- **View log**: When you click **View log**, the log details of the report are displayed.
- **View parameters**: When you click **View parameters**, the **VIEW PARAMETERS OF EXECUTION** window opens and displays the details used to run the report.
- **Share**: When you click **Share**, the URL is copied so you can share the report.
- **Results**: When you click **Results**, the **Pipeline snapshot report** page is displayed. You must then click **Export to CSV** to export the pipeline snapshot report.

A sample pipeline snapshot report page is shown as follows:



Releasing Software

A release is a repeatable plan that is used to drive deployments.

Releases are the organizing focus of software deliveries. Releases can represent any software-development-related event, from major milestones in the life of the company to comparatively minor events, like recurring maintenance. You can schedule a release at any point in your development-operations lifecycle. You can automate release schedules or manually run deployments.

A release contains deployment plans and events that are related in some business-meaningful way. For example, the deployment plans might represent the phases in a software development lifecycle, such as development, QA, and production. An event might represent a blackout that affects the release.

Every value stream has a corresponding pipeline generated for it, and every pipeline has deployment plan templates. The tight relationship between pipelines and deployments makes it easy to use pipelines to drive deployments. You can start by creating a release and adding deployment plans and events to it. Or you can start with value streams and pipelines and then join them to a release. You can run deployments directly from a pipeline or from a deployment plan.

You can coordinate several releases into a single event, which ensures that all release stakeholders can participate. If you add events to a release, you can use the calendar to organize releases and deployment plans.

Managing a release

Use the release details page to view details about a specific release.

This page contains the calendar and list of tasks. When viewing tasks you can select to view only your tasks or all tasks.

You can also use this page to access and work with release templates.

Use this page to view and work with archived releases removed from the calendar and list of tasks.

When you access a release from the **Releases** page, you can create a deployment plan and display overview details about the release, such as the deployment plans associated with the release with related details. From the **Overview** section of the **Releases** page, you can view the following release details for specific releases:

- **Plans** displays the plans associated with the release and task status, team, and date that the task is scheduled. You can access additional details for specific plans.
- **Task progress** displays the associated tasks that are not completed and contains information such as type of task, task owner, start and end time, duration, and task dependencies.
- **Approvals** list the approvals with the status of the approval and approver.
- **Change requests** displays a list of ServiceNow™ change requests.

The **Properties** section displays properties defined in the plan and property value.

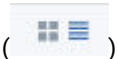
Managing releases

Release management is the process of planning, scheduling, and controlling software deliveries.

About this task

The **Releases** page displays information about releases. This page provides access to the calendar, tasks, events, release activities, and related details.

The page contains the calendar or list of tasks. Click **Calendar** to view the calendar. To hide the calender and task bar, click . To view tasks, click **Tasks**. When viewing tasks, you can select to view only your tasks or all tasks. To view events, click **Events**. To create an event, click .

Release activities are displayed in a tile or list () format. Using the search bar on the top of the page, you can limit the amount of information displayed by typing a filter to narrow the search results. Additionally, click the filter ( ) to limit the list of displayed release activities to: **Complete**, **In progress**, or **Scheduled**. Click an item in the **Release activities** list to view specific release details.

Click the tag ( ) to toggle on or off the display of tags associated with the release activity.

To create a new release, click **Create release**.

To import the releases and events, click **Import Release**.

You can also use this page to access and work with release templates by clicking **Templates** on the page.

Click **Archived** to view and work with releases removed from the **Releases** page and the calendar.

You can view release details, such as deployment plans associated with the release, by clicking **Overview**. This page will display headings that include: **Plans**, **Task progress**, **Approvals**, and **Change requests**.

- Plans associated with the release an be viewed by clicking **Plans**. The number above the **Plans** heading is the total number deployment plans related to this release.
- Tasks details can be viewed by clicking **Task progress**. The number above the **Task progress** heading displays the number of associated tasks not completed and total number of tasks. Click the task for additional task information.
- Click **Approvals** to display approvals details. The number above the **Approvals** heading is the total number of approvals. Click the approval name for details about a specific approval.
- Click **Change requests** to display a list of ServiceNow™ change requests. The number above the Change requests heading is the total number of ServiceNow™ change requests.

Click **Properties** to display a properties defined in the plan and property value. You can add or change the property value by double-clicking the **VALUE** property field. Click the check mark to save the value.

To create a new deployment plan, click **Create deployment plan**.

Creating releases

1. On the **Releases** page, click **Create release**
2. In the **Create release** window, type a name for the release in the **Name** field.
   This field can be referenced with the `${sys:release.name}` system property. A name is required.
3. In the **Tags** list, select one or more events or tags. You can also create a tag by typing the name of the new tag in the field.
   If you want the event to appear on the calendar, select an event that is created on the **Configure Calendar** page. Tags that you create with the **Create release** window are not displayed on the calendar, which can prevent calendar clutter.
4. In the **Description** field, type a description of the release. This field can be referenced with the `${sys:release.description}` system property.
5. In the **Deployment plan templates** field, select one or more templates to add to the release.
   Each template that you select, adds a deployment plan to the release with the name **Copy of *template_name***. You can convert any deployment plan into a template. If the selected templates have user-defined properties, you are prompted to provide values for them when you save the release.
6. In the **Start Time** field, select a starting date and time. This field can be referenced with the `${sys:release.start}` system property. Starting times are required for all releases.
7. In the **End Time** field, select the ending date and time. This field can be referenced with the `${sys:release.end}` system property.
8. In the **Team** list, select a team to manage the release. You can add deployment plans from other teams to the release event, not just the team that manages the release. All the teams that you belong to are available. A team is required.
9. If the selected templates have properties, click **Next**, and then provide values for the properties. Otherwise, click **Save** and view details to save the release and open the **Releases** details page. If you added templates to the release, a deployment plan for each template is created and displayed. If you didn't select any templates, an empty deployment plan is added to the release with the name **Plan: *release_name***.

   To display the release in the **Activities** list, ensure that the date range includes dates within the release's start or end times.

What to do next

After the release is created, you can add deployment plans, events, and tags to it.

Archiving releases

You can archive a release to remove it from the list of releases.

About this task

Although you cannot delete a release, you can store it in the archives. Archived releases are not displayed on the **Releases** page.

To archive a release, click the overflow ( ) and then select **Archive**.

Repeat this action for each release to be archived.

You can view archived releases and restore them by clicking **Archived** on the **Releases** page.

Events and tags

An event represents a release-related activity that you manage with the calendar. A tag is a label that you can add to a release or an event. You can use tags to filter the `Activities` list on the **Releases** page. Tags do not appear on the calendar.

Events are organized into general categories that include releases, holidays, and maintenance windows. You can add events to releases or use them independent of releases. When an event is assigned, it is represented on the calendar by an icon appropriate for the event category. You can use the icons to filter the **Releases** page and select releases for any affected date or range of dates.

Tags are assigned to releases and can be used to filter the **Releases** page. Assigned tags appear in Releases page but they do not appear on the calendar. You can use tags to avoid calendar clutter.

Restoring archived releases, plans, and templates

You can restore archived releases, deployment plans, and templates.

About this task

A restored release retains all of its original deployment plans.

1. On the **Releases** page, click **Archived**.
   The **Archive** page lists all archived items. The **TYPE** column displays whether the item is a release, plan, or template.
2. Locate the item to be archived.
   Using the search bar, you can limit the amount of information displayed by typing a filter to narrow the archive search results. Filters can contain keywords, teams, tags, or name. You can also click on some of the table headers to sort the table in ascending or descending order based on the heading selected. For example, clicking **ARCHIVED DATE**, list the items from earliest archive date to current. Click the tag ( ) to toggle on or off the displaying of tags associated with the release activity.
3. If an item needs to be restored, click the overflow ( ) and then select **Restore**.
   Repeat this step for each release to be restored.
4. Click **Releases** to return to the **Releases** page.

Results

- Restored releases are displayed in the list of releases.
- Restored releases retains all of its original deployment plans.
- Restored deployment plans have a status of **draft**.

Deployment plans

A deployment plan is a container for tasks. Tasks define the activities that your team runs to complete a software deployment.

You create a deployment plan by assigning a team to manage it and then adding tasks to the plan. Team members add tasks to the plan and run the tasks during deployments. The managing team also determines which HCL DevOps Deploy (Deploy) applications are available. After an integration is configured, all applications that are managed by the team in Deploy are available.

When you use the Pipeline feature, releases and deployment plans are automatically generated as you add stages and applications to the pipeline. You can start deployments directly from the pipeline, or open the deployment plan and start the deployment by running an eligible task.

The tasks define the activities that the team performs to complete a deployment. You can create new tasks, copy tasks from other deployment plans or import tasks from CSV files. When you are satisfied with the plan, you can schedule a deployment. You can also run a deployment at any time by starting one of the plan's tasks.

Tasks are displayed on individual rows in deployment plans. Each row contains information that identifies the task and provides its status. Each row also contains action icons that are used during deployments, such as **Start** task or **Skip**.

You can run the tasks sequentially or parallely beginning with the first task, or topmost task, in the plan. After the first task finishes, the second task is eligible to run, and so forth. The order in which tasks are eligible to run is called the execution order.

The execution order can be modified by combining tasks into groups. When you create a group on page 1849, you set the group's execution pattern, which can be sequential or parallel. If a group has a sequential execution pattern, its tasks run sequentially starting with the first task. If a group has a parallel execution pattern, tasks in the group can be started in any order and they can run simultaneously. If a plan consists entirely of parallel tasks, you can start any task regardless of its position in the list.

Task eligibility can also be effected by task dependencies. If a task has any prerequisites on page 1848, it cannot start, even if it is otherwise eligible, until all prerequisite tasks are resolved.

Some tasks start automatically as soon as they are eligible to run, while other tasks are started manually. Whether a task starts automatically or not depends on its type. Manual tasks, as the name implies, are started manually. Deploy and Delayed-type tasks automatically start as soon as they are eligible.

Eligible tasks display a **Start Task** button. To start a deployment, you start one of a plan's eligible tasks. Scheduled deployments start automatically at the scheduled time if an eligible task is of a type that can start automatically, such as Deploy.

Managing deployment plans

You can change existing deployment plans and view current status from the **Deployment Plan** details page.

About this task

The **Deployment Plan** details page displays a list of tasks, properties, versions, and the change history associated with the plan.

- Click **Tasks** to work with the tasks in the plan. You can add tasks, schedule tasks, skip a task, and reopen a completed task.
- Click the **Properties** to view properties and their value.
- Click the **Versions** to view versions.
- Click the **Change history** to view a list of the changes, revision number, user who made the revision, and date. You can also restore a plan to a specific version by clicking restore ( ).

From this page you can modify and archive the plan, change the name of the plan, make a copy of the plan, and create a template from the plan. Access these features from the overflow menu ( ) at the top right corner of the page.

- Click **Import tasks** to incorporate tasks into the deployment plan from various file sources including spreadsheets, project management tools, or IBM® DevOps Release.
- Click **Estimate task times** for approximating task durations.
- Click **Edit** to modify the deployment plan. With this option, you can add events and tags to the release event.
- Click **Copy** to copy a deployment plan. A copy of the selected plan is added to the release.
- Click **Copy as template** to create a template based on this plan.
- Click **Archive plan** to remove the deployment plan from the release. Removed plans are not permanently deleted and can be added to another release.

Creating deployment plans

You can start with a blank plan that has no defined tasks, copy an existing plan, or start with a plan template. You can also auto-create them by assigning applications to pipelines. When you copy a plan or use a template, many tasks are already defined.

About this task

Complete the following steps to create a deployment plan:

1. On the **Releases** details page, click **Create Deployment Plan**.
2. On the Create Deployment Plan window, in the **Deployment plan template** list, specify the template that you want to base the plan on. The default value is **None**.
3. In the **Plan name** field, enter the plan name.
4. In the **Description** field, enter the description.
5. In the **Team** list, select a team to manage the plan. All teams that you are a member of are available. Team members can modify the plan and manage tasks. The HCL DevOps Deploy applications that are managed by the team in Deploy are available to assign to Deploy type tasks.

6. To schedule a deployment for the plan, click **Start time**, and then select a date and time for the deployment. Scheduled deployments automatically start at the scheduled time if the plan contains eligible auto tasks. You can schedule a deployment for any future time.
7. In the **Tags** list, select tags that you want to add to the plan.
8. Click **Save**. If you created the plan on a **Releases** details page, the plan is part of the selected release. After you save the plan, click **Edit** to modify plan details.

Note: You can create one deployment plan as a primary linked to many secondary deployment plans. Deployment is executed only from primary deployment plan.

Importing tasks into deployment plans

You can import tasks that are defined in comma-separated values (CSV) files into a deployment plan.

About this task

You can import tasks from CSV files that are exported from IBM® DevOps Release or another application capable of creating CSV files. When you import a deployment plan from Release, segments are converted into groups with the same execution pattern as the original segments. Segment tasks are bracketed by note-type Start Segment and End Segment tasks. Task dependencies are preserved during import. Segment dependencies are represented by dependencies to End Segment tasks.

To learn about deployment plans exported from Release, download the `SamplePlan.csv` file from the Import Tasks dialog.

Note: When you import tasks, tasks that are already defined in the deployment plan are overwritten and replaced by the tasks in the CSV file.

The following table contains the fields that define a task in an Release deployment plan. The first row in the CSV file contains the column headers, which correspond to the table's Field column. The rows after the first row contain task data, one task per row. The order of the columns does not matter.

Note: If you are creating tasks in a CSV file, you must include the required fields that are identified by the asterisk character (*) in the following table.

Table 1182. Release deployment plan fields

Field	Description
segmentName*	The name segment. Required field.
name*	The name of the deployment task. Required field.
type*	Task type. Possible values are note, waitfortimetask, ucdtask, or manual. During import, if the field contains an unrecognized value, the task is assigned the manual type.
description	Description of the deployment plan.

Table 1182. Release deployment plan fields (continued)

Field	Description
property:processId	The process ID in HCL DevOps Deploy (Deploy) that corresponds to the task.
property:task-environments	Application environments available to the task.
property:mailRecipients	Email addresses of users who receive email messages when notifications are sent.
taskTagNames	The names of the tags that are associated with the task.
taskPrequisiteIds	A comma-separated list of IDs for tasks that must complete before this task can start.
segmentPrerequisiteIds	A comma-separated list of IDs for segments with which the current segment has dependencies.
taskPrequisiteNames	A comma-separated list of names for tasks that must complete before this task can start.
segmentPrerequisiteNames	A comma-separated list of names for segments with which the current segment has dependencies.
assignedUserEmail	Email address of the user who is assigned to the current task.
executorGroup	User group assigned to the current task.
segmentPattern	Execution pattern of the selected segment.

Complete the following steps to import tasks into a deployment plan:

1. On the **Deployment Plan** details page, click **Import Tasks**.
2. In the Import Tasks dialog, click **Choose File**, and then select the CSV file in addition to confirming the choice in the dialog box. You can also drag a file onto the field.
3. Click **Import**. The tasks are added to the deployment plan. If the import process is unsuccessful, click **View Report** to display the error log.

Copying deployment plans

You can duplicate deployment plans by copying them.

About this task

The status of a copied or promoted plan is `draft` even if the original plan has a status of `done` or `in progress`.

1. **Optional:** To copy a plan, click the overflow icon ( ) and select **Copy plan** from the drop-down on the **Releases** details page. When you copy a plan, a new plan with the name "Copy of `plan_name`" is inserted into the **Releases** details page. The copied plan contains all the tasks that are defined in the original plan. If HCL DevOps Deploy (Deploy) tasks are in the plan, the applications, versions, and environments that are selected in the original are also selected in the copied plan. Applications, versions, and environments are displayed on the **Versions** tab on the Deployment Plan Detail page.
2. **Optional:** To create a template from a plan, click the overflow icon ( ) and select **Copy as template** from the drop-down on the **Releases** details page. When you copy a plan as a template, a new plan with the name "Copy of `plan_name`" is inserted into the **Templates** page. The copied plan contains all the tasks that are defined in the original plan. If Deploy tasks are in the plan, the applications, versions, and environments that are selected in the original are also selected in the template.

Managing deployment plan templates

You can convert deployment plans to create deployment plan templates.

About this task

To create a template, click the overflow icon ( ) and select **Copy as template** from the drop-down on the **Releases** details page. When you create a template, a new plan with the name "Copy of plan_name" is inserted into the **Templates** page. The status of a template is `template` even if the original plan has a status of `done` or `in progress`. The template contains all the tasks that are defined in the original plan. If HCL DevOps Deploy (Deploy) tasks are in the plan, the applications selected in the original are also selected in the template.

You cannot run deployments with templates. To use a template for a deployment, first copy or promote it and then use the copied or promoted plan.

Archiving deployment plans

You can remove deployment plans from releases by archiving them.

About this task

Deployment plans cannot be permanently deleted, but can be archived. When you archive a deployment plan, it has a status of **Archived** and is not displayed in the list of plans on a **Releases** details page.

You can view archived deployment plans and restore them by clicking **Archived** on the **Releases** page.

1. Click **Plan** for a release to view a list of associated deployment plans.
2. On the line of the plan to be archived, click the overflow menu icon ( ) and select **Archive this plan**.
   Repeat this step for each plan to be archived.

Running deployments

Complete deployments by resolving the tasks in a deployment plan.

A deployment starts when one of the plan's eligible tasks starts. A scheduled deployment starts automatically at the scheduled time if one of its eligible tasks is an auto task, such as HCL DevOps Deploy (Deploy) tasks. Otherwise, begin a deployment by starting one of the plan's eligible tasks. If the plan contains several eligible tasks, you can start any of them. You can manually start a deployment at any time. You can manually start a scheduled deployment before its scheduled time.

The deployment plan's execution pattern determines when tasks are eligible to start. If the plan has the sequential execution pattern, which is the default pattern, tasks become eligible in the order in which they are listed in the plan, starting with the first task. After the first task resolves, the second task becomes eligible. Start eligible manual tasks by clicking the task's **Start** button. Auto tasks, such as Deploy tasks, start automatically as soon as they become eligible.

The tasks in a group with a parallel execution pattern become eligible simultaneously. Tasks in parallel groups can be started in any order. Nested groups and tasks with ad hoc dependencies can affect the execution pattern.

You can modify a deployment plan after you start a deployment. You can add, delete, and modify tasks.

After all tasks are resolved, the deployment is complete. A task is resolved if it has a status of `Complete`, `Failed`, or `Skipped`. If you reopen a task or add a task after the deployment is complete, the deployment's status changes to `In Progress`.

Starting deployments manually

Starting deployments manually

About this task

You can start a deployment manually at any time, including deployments that are scheduled to start later.

Tasks in a deployment plan might be inapplicable for deployment. HCL DevOps Deploy (Deploy) tasks are inapplicable if no version or environment is specified for the tasks. When you create Deploy tasks, you can delay selecting the environment and version by using the **Use Version** option. Before the deployment starts, ensure that all Deploy tasks are applicable for the upcoming deployment.

You can deliberately exclude tasks from a deployment. You can make tagged tasks inapplicable by excluding their tags from the deployment. Tasks with excluded tags are inapplicable for deployment.

Inapplicable tasks have the status of `Not Applicable`. Tasks with the `Not Applicable` status cannot be started. If an inapplicable task is a prerequisite for an active task, the dependency is ignored.

To start a deployment, complete the following steps:

1. On the **Releases** details page, click the deployment plan that you want to use for the deployment. The **Deployment Plan** details page is displayed.
2. **Optional:** To make tagged tasks inapplicable for the current deployment, click **Versions**, and then in the **Tags** area, clear the tags. If a task has more than one tag, clear all of them.

3. To select the version or environment for any inapplicable Deploy tasks, click **Versions**, and then select the environment and version. You can also change your choices for tasks that have their environments and versions already selected.
4. **Optional:** Click **Hide Not Applicable Tasks** to remove inapplicable tasks from the displayed task list. Hidden tasks are not removed from the deployment plan.
5. Click the **Start** action for one of the plan's eligible tasks. If more than one task is eligible, you can start any of them. Only eligible tasks display the start button. A started task has the status of `In Progress` until it is resolved.
   Result
   After a deployment starts, the plan's status changes to `In Progress`. The status remains `In Progress` until all tasks are resolved. When all tasks are resolved, the plan's status changes to `Done`.

Resolving tasks

Complete a deployment by resolving the tasks in the deployment plan.

About this task

A started task is resolved when its status changes to `Complete`, `Failed`, or `Skipped`.

You resolve manual tasks by using the appropriate status action. Auto tasks, such as HCL DevOps Deploy (Deploy) tasks or Jenkins®, change status automatically as conditions change. However, you can manually resolve auto tasks. For example, you might manually fail an Deploy task that is taking too long to complete.

To resolve a started task, apply one of the following statuses:

- Click **Complete** to finish the task. `Complete` means that the task finished with the expected results.
- Click Skip to skip the task for the current deployment.
- Click **Fail** to end the task. `Fail` means that the task did not finish or finished with unexpected results.
- Click **Task Reopen** to open a resolved a task. If all tasks are completed, reopening a task reopens the deployment.

Importing tasks into deployment plans

You can import tasks that are defined in comma-separated values (CSV) files into a deployment plan.

About this task

You can import tasks from CSV files that are exported from IBM® DevOps Release or another application capable of creating CSV files. When you import a deployment plan from Release, segments are converted into groups with the same execution pattern as the original segments. Segment tasks are bracketed by note-type Start Segment and End Segment tasks. Task dependencies are preserved during import. Segment dependencies are represented by dependencies to End Segment tasks.

To learn about deployment plans exported from Release, download the `SamplePlan.csv` file from the Import Tasks dialog.

Note: When you import tasks, tasks that are already defined in the deployment plan are overwritten and replaced by the tasks in the CSV file.

The following table contains the fields that define a task in an Release deployment plan. The first row in the CSV file contains the column headers, which correspond to the table's Field column. The rows after the first row contain task data, one task per row. The order of the columns does not matter.

Note: If you are creating tasks in a CSV file, you must include the required fields that are identified by the asterisk character (*) in the following table.

Table 1183. Release deployment plan fields

Field	Description
segmentName*	The name segment. Required field.
name*	The name of the deployment task. Required field.
type*	Task type. Possible values are note, waitfortimetask, ucdtask, or manual. During import, if the field contains an unrecognized value, the task is assigned the manual type.
description	Description of the deployment plan.
property:processId	The process ID in HCL DevOps Deploy (Deploy) that corresponds to the task.
property:task-environments	Application environments available to the task.
property:mailRecipients	Email addresses of users who receive email messages when notifications are sent.
taskTagNames	The names of the tags that are associated with the task.
taskPrequisiteIds	A comma-separated list of IDs for tasks that must complete before this task can start.
segmentPrerequisiteIds	A comma-separated list of IDs for segments with which the current segment has dependencies.
taskPrequisiteNames	A comma-separated list of names for tasks that must complete before this task can start.
segmentPrerequisiteNames	A comma-separated list of names for segments with which the current segment has dependencies.
assignedUserEmail	Email address of the user who is assigned to the current task.
executorGroup	User group assigned to the current task.

Table 1183. Release deployment plan fields (continued)

Field	Description
segmentPattern	Execution pattern of the selected segment.

Complete the following steps to import tasks into a deployment plan:

1. On the **Deployment Plan** details page, click **Import Tasks**.
2. In the Import Tasks dialog, click **Choose File**, and then select the CSV file in addition to confirming the choice in the dialog box. You can also drag a file onto the field.
3. Click **Import**. The tasks are added to the deployment plan. If the import process is unsuccessful, click **View Report** to display the error log.

Running scheduled deployments

A scheduled deployment starts automatically at its scheduled time if any of its auto tasks are eligible to start.

About this task

If a deployment plan does not contain any eligible auto tasks, start the deployment by manually starting one of the eligible tasks. If the deployment plan belongs to a pipeline-generated release, you can start the deployment directly from the pipeline.

You can manually start a scheduled deployment at any time even if the plan has eligible auto tasks.

Tasks

A task represents a business-meaningful activity that is associated with a software deployment. Each task is defined on a separate row in a deployment plan. During a deployment, you run tasks defined in the deployment plan to complete the deployment.

Most tasks have a starting and ending point, and a measurable duration. A task can have other attributes depending on its type. A task can be of one of following types:

Ansible

Automation tool used for configuration management, application deployment, intraservice orchestration, and provisioning tasks.

Delayed

Represents a critical event that happens at a specific time.

Email

Sends an email message.

Header/Note

an organizational element. For example, you might use a header task to identify a task group.

Jenkins®

Runs a Jenkins® job or pipeline.

Manual

Represents any activity that is associated with a software deployment, such as taking a server offline or updating a database. Users assigned manual tasks can be sent email notifications when their tasks start or fail.

Run a Shell Script

Shell script tasks are used to execute a sequence of commands. The shell is the command line interface (CLI) for the operating system (OS) used to interpret the sequence of commands communicating with the software interface.

Run another plan

Runs deployments for other deployment plans participating in the same release event. This is an experimental task type.

ServiceNow™

Creates and manages ServiceNow tickets.

HCL DevOps Deploy (Deploy) (requires Deploy integration)

Use this task to run an Deploy application process.

Deploy Status (requires Deploy integration)

Use this task to add a status to all component versions contained within an Deploy snapshot.

Wait for approval

Pauses a deployment until the designated users approve the task.

You can add tasks to deployment plans by one of the following methods:

- Create a new task
- Import from CSV files that are created by Deploy or another application
- Copy tasks from other deployment plans

Additionally, tasks are auto-created when you add an application to a pipeline on page 3127.

Managing tasks

A list of associated tasks can be viewed from a **Deployment Plan** details page. The list of tasks include details about the task. You use action icons to change the task's status during a deployment, such as start or skip a task.

About this task

Each task in a deployment plan is contained in a separate row. Click ( ) to show or hide associated tasks from the list.

Click ( ) show or hide tags.

The information that is displayed for each task is described in the following table.

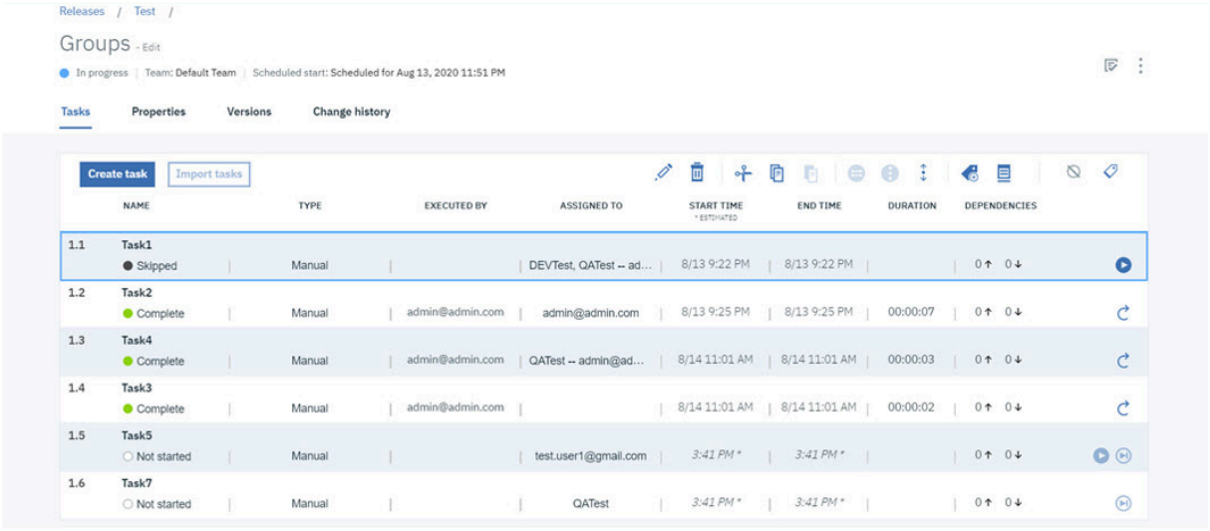


Table 1184. Task properties

Property	Description
Name	Task name
Status	Task status: Not started, complete, failed, skipped, not applicable
Type	Type of task: Delayed, Deployment Risk Analysis, Email, Header/Note, Jenkins®, Manual, Run another plan, ServiceNow™, HCL DevOps Deploy, and Wait for approval.
Executed by	Person who executes the manual task
Assigned to	Groups and users designated as the task owner
Start time	Start time or expected start time based on scheduled start, or estimated duration of other tasks
End time	Time that the task resolved
Duration	Length of time in minutes from task start to task resolution.

Table 1184. Task properties (continued)

Property	Description
Dependencies	Indicates the number of tasks that are prerequisites for the task, and dependent on the task
Action icons	One or more actions used to change the status of the task during a deployment

Tasks are listed in the order that they run. The order of the tasks can be changed by selecting one or more tasks and dragging to the new location.

At the end of each row are applicable actions associated with the task. Depending on the task you can perform one or more of the following actions.

- Start task ( )
- Skip task ( )
- Re-run task ( )
- Complete task ( )
- Fail task ( )

You can manage the tasks in a deployment plan from the task bar above the table. When a task is selected, the task bar displays and you can perform the following actions:

- Click ( ) to edit the task.
- To copy a task or group, select one or more tasks and click , and then place the cursor where you want to insert the copied task and click to paste the selection.
- To cut a task or group from a deployment plan, select the task and click .
- To delete a task, select the task and click . The task is removed from the deployment plan.

- To create a group of tasks that can run simultaneously, select the tasks and click . The group must consist of at least two tasks.
- To create a group of tasks that are to consecutively run, select the tasks and click . This is the default execution pattern.
- To add or remove tasks that must complete first, select the task and click . In the dialog box, select or type the prerequisite task to be added. Click **X** next to the prerequisite to be removed. When finished, click **Save**.
- To add or remove tags, select the task and click . In the dialog box, type or select tags to add. To remove a tag, click **X** next to the tag to be removed. When finished, click **Save**.
- To view information about when the task started, completed, and errors, click .

Creating tasks

By default, new tasks are inserted at the bottom of the deployment plan. After a task is created, you can move it, or copy it and paste it into another deployment plan. You can also create dependencies with other tasks on page 1848.

About this task

Tasks can be added to a deployment by creating a new task or importing an existing task.

After you save a task, actions are displayed for the task. You use actions to change the task's status during a deployment.

All tasks have the skip action. Other actions, such as the start action, are displayed when the context is appropriate for them.

1. From the **Deployment Plan** details page, click **Create task** to create one or more tasks.
2. Select a task type.

   Note: The availability of task types will depend on your version of HCL DevOps Deploy (Deploy) and integrations.

   Possible task types include:
   - Ansible
   - Delayed
   - Deployment Risk Analysis
   - Email
   - Header / Note
   - Jenkins

- Manual
- Run a Shell Script (requires shell script plugin)
- Run another plan
- ServiceNow™
- Deploy (requires Deploy integration)
- Deploy Status (requires Deploy integration)
- Wait for approval

3. Provide information in required and optional fields displayed according to the task type selected.
4. Click **Save and create another** to create additional tasks. Click **Save** when finished.

Managing task dependencies

You can make a task a prerequisite for other tasks. If a task is a prerequisite, dependent tasks cannot start, even if they are otherwise eligible, until the prerequisite task is resolved.

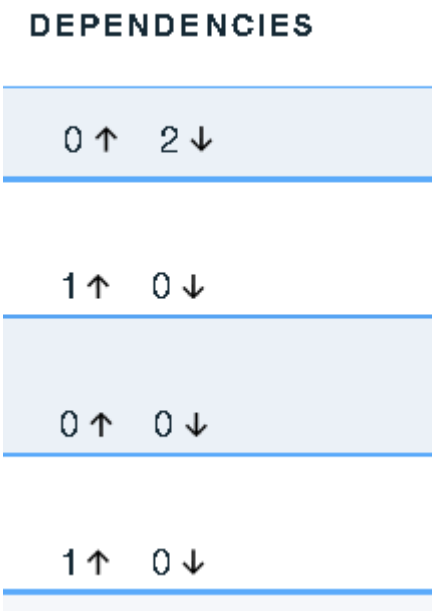
About this task

A task can have multiple dependent tasks and multiple prerequisite tasks. You can define dependencies for a task with any other task in the deployment plan. However, you cannot create circular dependencies. You cannot, for example, make a task dependent on a task that itself depends on the first task. By controlling task dependencies, you can ensure that events occur in their expected order.

To make a task a prerequisite for other tasks, complete the following steps:

1. On the **Deployment Plan** details page, select a task or task group, and then click . You can select multiple tasks and groups.
2. In the **Manage prerequisites for selected tasks** dialog box, in the **Prerequisite tasks for selected tasks** drop-down list, select the prerequisite task.
3. Click **Save**.
   Result
   Task dependencies are shown in the **DEPENDENCIES** column on the **Deployment Plan** details page. Up arrows indicate task prerequisites; down arrows indicate task dependencies. In the following figure, the first task does not

have any prerequisites and has two tasks dependent on it. The second task has one prerequisite task and there are no tasks dependent on it.



What to do next

To review or modify dependencies, select the task and then click . Use the **Manage prerequisites for selected tasks** dialog box to modify or remove dependencies.

Creating task groups

You can combine two or more tasks into a task group. When you create a group, you define the whether the group task are executed sequentially or in parallel.

About this task

Tasks can run in a parallel or sequential pattern. When you create the task, you decide the execution pattern. Tasks without dependencies can execute in any order or simultaneously which is *parallel* execution. Tasks that must execute in specific order are *sequential* execution. Tasks are done in list-order starting with the first or top-most task.

You can embed groups within other groups. A sequential-pattern group can be embedded within a parallel-pattern group, and vice versa; however, you cannot embed a sequential-pattern group within another sequential group, or a parallel-pattern group within another parallel group.

1. On the **Deployment Plan** details page, select two or more tasks.
2. Depending on the type of group you want to create, complete one of the following.
   Choose from:

- To create a parallel group, click the Parallel icon)). If you cannot create a parallel group with the selected tasks, the icon is disabled. For example, you cannot create a parallel group if all the selected tasks are already in a parallel group.
- To create a sequential group, click the Sequential icon ( ).

Result

The group is formed and a group select bar is added to the deployment plan. If you selected discontiguous tasks, the tasks form a contiguous list starting with the topmost selected task.

Managing task groups

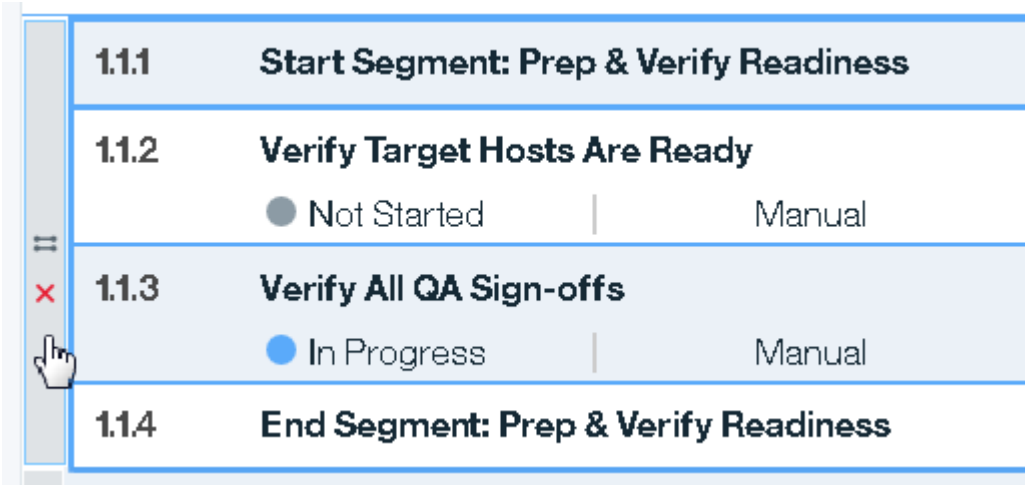
You can combine two or more tasks into a task group. When you create a group, you define the whether the group task are executed sequentially or in parallel.

About this task

You can recognize task groups in the list of task, there is a bar connects the grouped tasks and contains an icon to identify if it is a parallel or sequential group.

- parallel
- sequential

The following image shows a parallel group.



You can manage task groups in several ways:

- To move a group, select the group select bar or click anywhere on the group, and then drag it to a new location.
- To copy a group, select the group and click , and then place the cursor where you want to insert the copied group and click .
- To cut a group, select the group and click .
- To ungroup a group, select the group and click the **Ungroup** icon on the group select bar.
- To delete the tasks in a group, select the group and click . The tasks are removed from the deployment plan.

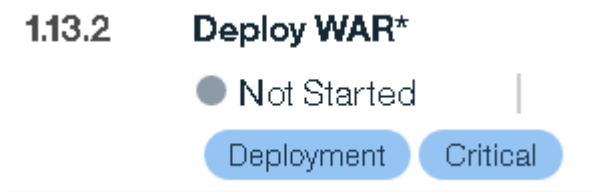
Managing task tags

Tags are organizing elements that you can add to tasks. You can filter deployment plans by tag. For example, during a deployment to a production environment, you might disable tasks with the `DEV_only` tag, which is intended to be used only in development environments.

About this task

To add a tag to a task, complete the following steps:

1. On the **Deployment Plan** details page, select a task or task group, and then click . You can select multiple tasks and groups.
2. In the **Manage Tags for Selected Tasks** dialog box, in the **Tags** drop-down list, select tags. You can create a new tag by typing a name in the list's text box.
3. Click **Save**.
   Tags are displayed on the task rows in the **Deployment Plan** details page. In the following figure, the Deploy WAR task has two tags assigned to it, `Deployment`, and `Critical`.



   The tags used by a deployment plan are displayed on the **Deployment Plan** details page by clicking **Versions**. To render tasks with a specific tag `Not Applicable` for a deployment, clear the tag. Tasks with the `Not Applicable` status cannot be started.

Creating run another plan tasks

A run another plan task runs a deployment for another deployment plan.

About this task

The targeted plan can be created from a template or a tag. If you use a template, the template and the targeted plan must be in the same release as the run another plan task.

To create a run another plan task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create task**. If you want to insert a task at a specific position in the plan, select a task before you use the **Create task** action. The new task is inserted before the selected task.
2. In the **Create task** dialog box, in the **Type** drop-down list, select **Run another plan**.
3. In the **Name** field, enter a name for the task.
4. In the **How do you want to choose another plan** area, select an option.
   If you choose, **By template**, the template and associated deployment plan must be in the same release as the task's parent plan. If you choose, **By tag**, the tagged deployment plan must be in the same release as the task's parent plan as demonstrated in this example on page 1852 of creating a primary and secondary template.
5. In the **Run plans created from this template** list, select a deployment plan template. The template and associated deployment plan must be in the same release as the task's parent plan.
6. In the **Run plans with this tag** list, select the tag associated with the template to the task.
   You can select multiple tags. To create a tag, type the tag name in text box.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When the task runs, a deployment starts for the deployment plan that was created from the selected template or tagged plan. If multiple plans in the release event are eligible, deployments start for all of them. While the other deployments run, status information is provided by the run another plan task. Expand the task to see the status information. You can open the other deployments by using links on the task. While the other deployments run, the run another plan task has a status of **In Progress**.

Creating run another plan tasks: Example

A run another plan task runs a deployment for another deployment plan.

About this task

This example is a targeted plan that is created from a tag. The tagged deployment plan must be in the same release as the task's parent plan.

To create a targeted plan that is created from a tag, complete the following steps:

1. Create the primary template.
    a. On the **Releases** page, click **Templates > Create template**.
    b. In the **Plan template name** field, enter the name for the primary template.
    Accept the default values for the remaining fields.
    c. Click **Save**.
    d. Click the name of the primary template.
2. Referring to the Creating tasks on page 1847 page, create independent and dependent tasks for the primary template by using field values that are detailed in the following tables for each task:

Table 1185. Manual task

Create task field	Value
Type	Select **Manual**.
Name	Enter the name for the manual task.
Description	Default value.
Duration (minutes)	Default value.
Tags	Default value.
Assigned groups and users	Default value.

Table 1186. Run another plan task

Create task field	Value
Type	Select **Run another plan**.
Name	Enter the name for the secondary task.
How do you want to choose another plan?	**By tag**
Run plans with this tag	Select a tag or create a new tag by typing a name in the field. The primary deployment plan will find any other plan with this tag name when it runs.
Tags	Default value.

3. Create the secondary template.
    a. On the **Releases** page, click **Templates > Create template**.
    b. In the **Plan template name** field, enter the name for the secondary template.
    Accept the default values for the remaining fields.

c. Click **Save**.

d. Click the name of the secondary template.

4. Referring to the Creating tasks on page 1847 page, create independent and dependent tasks for the secondary template by using field values that are detailed in the following tables for each task:

Table 1187. Wait for approval task

Create task field	Value
Type	**Wait for approval** or any task that starts automatically.
Name	Enter the name for the task.
Approval prompt	Default value or enter a prompt message.
Approvers	Select the name of the approver. You can select multiple approvers.
Required Approver	Select **Any approver can approve** or **All approvers must approve**

Table 1188. Manual task

Create task field	Value
Type	**Manual**
Name	Enter the name for the task.
Description	Default value.
Duration (minutes)	Default value.
Tags	Default value.
Assigned groups and users	Default value.

5. Referring to the Creating releases on page 1833 page, create a release with the primary and secondary templates by using field values that are detailed in the following table:

Table 1189. Create release

Create release field	Value
Name	Enter the name for the release.
Tags	Select the name for the required tag.
Description	Default value.
Deployment plan templates	Select the name of the primary template that you created.

Create release field	Value
Start time	Default value.
End time	Default value.
Team	Default value.

6. On the **Releases** page, click the name of the release.
7. Click the primary plan to display the independent task and dependent task with a tag in the template.
8. At the top of the page, click the link to go back to the release.
9. Create the deployment plan.
    a. Click **Create deployment plan**.
    b. In the **Deployment plan template** list, select the name of the secondary template that you created.
    c. In the **Tags** field, select the name for the required tag.
    d. Click **Save**.

    Note: You can create one deployment plan as a primary linked to many secondary deployment plans. Deployment is executed only from primary deployment plan.

10. Click the primary plan to display the independent and dependent tasks.
    The dependent task is now ready to run with the independent task being a manual type.

    a. On the manual task, click **Start task** to start the task.

    b. On the manual task, click **Approve task** to complete the task.
    The manual task displays the status as `Complete`, and the dependent task displays `In progress`.

    c. On the dependent task, click the arrow to the left of **In progress**.

    d. Click **Go to** to go to the dependent task.

    e. On the "Wait for approval" task, click **Approve task** to approve the task.

    f. On the "Manual" task, click **Start task** to start the task.

    g. On the "Manual" task, click **Complete task** to complete the task.

    h. At the top of the page, click the link to go back to the release.

The **Manual** and **Run another plan** tasks are both displayed as completed in the secondary plan.

i. At the top of the page, click the link to go back to the primary plan to display the plans in the release that are completed.

Creating delay tasks

Delayed-type tasks represent milestones or critical events during a deployment. With delayed tasks, you can ensure that important tasks start at the expected time. Typically, delayed tasks are prerequisites for other important tasks.

About this task

A delayed task starts as soon as it is eligible to run, and it finishes at a user-specified time. A started delayed-type task has a status of `In Progress` until it reaches its planned-for time, when it then changes its status to `Complete`. When a delayed tasks completes, auto tasks that are dependent on it start running.

Complete the following tasks to create a delayed task:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Delayed**.
3. In the **Name** field, enter a name for the task.
4. In the **Delay type** field, select the method used to determine the length of the delay.
   If you select **Delay by selecting a time and time zone**, use the **Delay until time** and **Time zone** fields to specify when the task will complete. If you select **Delay by using a date and time property**, use the **Delay property** field to enter a property name. The property value can be set at run-time when the deployment starts. Refer to the Properties reference for information about properties:

   ```
   /docs/services/UCCR/UCCR_property_ref.html#property_overview
   ```

   Otherwise, if you select **Delay by selecting a number of minutes**, use the **Minutes** field and enter the number of minutes between tasks.
5. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating email tasks

An email task sends an email message when the task runs. You specify the email's recipients and message when you create the task. Email tasks are auto tasks and run as soon as they are eligible.

Before you begin

Before using email tasks in your deployments, setup an email server. on page 1800

About this task

To create an email task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Email**.
3. In the **Name** field, enter a name for the task.
4. In the **Recipients** field, enter or select the email's recipient.
   The list of available recipients includes the users and groups who are members the team that manages the task. You can also type the email addresses of non-team members. You can specify multiple recipients.
5. In the **Email subject** field, enter the email topic.
   You can use properties in this field.
6. In the **Email message** field, enter or paste the email message.
   You can use properties in this field. For example, you might notify email recipients that a release is started by using a message similar to this one, "The `` `${sys:release.name}` `` release has started." See the Properties reference for information about properties on page 1871.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When the task runs, the recipients receive an email with the subject you specified when you created the task.

Creating header/note tasks

Header tasks represent organization elements that you can add to deployment plans. If you create a task group, you might identify the group with a header task. Header tasks can have dependencies like any other task.

About this task

You can organize task groups with header/note tasks. You can create a note by adding information about a group or task. When you import a deployment plan from IBM® DevOps Release, segment tasks are bracketed by note-type Start Segment and End Segment tasks. Segment dependencies are represented by dependencies to End Segment tasks.

To create a header task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Header / Note**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter or paste a description.
   You might enter a note, a reminder, or other instructions in the field.
5. Click **Save**.

Result

The task is inserted into the deployment plan.

Creating Jenkins (Legacy) tasks

A Jenkins (Legacy) task runs a job in your integrated Jenkins instance.

Before you begin

Before using Jenkins (Legacy) tasks in your deployments, add the plugin to your Jenkins instance and configure an integration. on page 1886 After integrating Measure to Jenkins, you can add Jenkins jobs to you pipelines and releases.

About this task

To create a Jenkins (Legacy) task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Jenkins (Legacy)**.
3. In the **Name** field, enter a name for the task.
4. In the **Jenkins Job or Pipeline** field, select a job or pipeline.
5. In the **Duration (minutes)** field, enter an estimate for the length of the task.
6. In the **Tags** list, select an item to attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating manual tasks

Manual tasks represent some activity that is associated with a software release. Manual tasks have a start point, end point, and a measurable duration.

About this task

To create a manual task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Manual**.
3. In the **Name** field, enter a name for the task.
4. In the **Duration (minutes)** field, enter the number of minutes that you expect the task to run until it is completed.
   The estimated duration is used to calculate expected deployment times.
5. In the **Tags** list, select an item to attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.

6. In the **Assigned groups and users** list, select the task owner.
7. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

Once the manual task is executed, you can select the task on the deployment plan details page on page 1845, click to display the **Edit task** window, and view who executed the task in the **Executed By** field.

Creating run shell script tasks

Shell script tasks are used to execute a sequence of commands.

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create task** dialog box, in the **Type** list, select **Run a Shell Script Task**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter or paste a description.
   You might enter a note, a reminder, or other instructions in the field.
5. In the **Shell Script** field, enter or paste the shell script code. An example shell script code is the following:

```
echo "Hello World"
sleep 2
"Do some stuff..."
sleep 2
echo "Task finished"
```

6. Click **Save**.
   Result
   The task is inserted into the deployment plan.

Creating ServiceNow tasks

A ServiceNow™ task can create and update ServiceNow™ change requests.

Before you begin

ServiceNow task prerequisites:

- ServiceNow tasks work with Internet-accessible instances of ServiceNow, such as `my_instance.service-now.com`.
- ServiceNow tasks work with the Jakarta® release of ServiceNow, and references the current API version.
- The user supplying credentials for the task must have access to the ServiceNow API, and permission to see and modify all change request fields. At a minimum, the user must be in a ServiceNow role that has the Create Change permission.

About this task

A ServiceNow task can perform one of the following actions:

- **Create** creates a ServiceNow change request. You can set the values for all request fields, and pass the request ID to update, and wait tasks.
- **Update** can update the values for any request field.
- **Wait** waits for a particular value in a specific field. The field can be any field on the change request.

A ServiceNow task can include ServiceNow properties and Measure properties. ServiceNow properties have the following syntax: `propertyName=value`.

This short video describes how to use ServiceNow with Measure.

To create a ServiceNow task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **ServiceNow**.
3. In the **Name** field, enter a name for the task.
4. In the **Tags** list, attach a tag to the task.
   You can select multiple tags. To create a tag, type the tag name in list's text field.
5. In the **Connection settings** area, enter the URL of the ServiceNow instance, and the user ID, and password of the ServiceNow user.
6. In the **Action** list, select the action type for the task.
   The action types are defined in this list:
   - The **create** action creates a ServiceNow change request with the properties defined in the task.
   - The **update** action modifies properties in change request.
   - The **wait** action waits for a specific value in a change request field.
7. Complete the action-specific parameters, as described in the following tables:

Table 1190. Create action

Create action	Description
Request type	You can use the ServiceNow default types or a user-created type. Default types are normal, emergency, and standard. Normal requests pass through all stages

Create action	Description
	of the ServiceNow state model. Standard types are pre-authorized and skip the Assess and Authorize states. Emergency types skip the Assess state.
Short description	The text you enter is displayed in the Short description field of the ServiceNow change request.
Assignment group	The ServiceNow agent or group responsible for resolving the change request. Possible assignees include agents and groups available to the ServiceNow instance.
Additional properties	The ServiceNow properties set by this task. Each property must be on its own line. For example, a simple request might include the following fragments: `approval=requested`.
Output property	Measure property that contains the change request ID. For example, if you specify `service_now_stage_ticket_id`, you can use the property with update and wait actions to identify the ticket.

Table 1191. Update actions

Update action	Description
System ID	The ID of the ServiceNow change request. If you capture the ID with a create action **Output property** field, you can use it here. For example, if you specify `service_now_stage_ticket_id` in the **Output property** field of a create action, you can use the ID by specifying `${service_now_stage_ticket_id}`. See the Properties reference for information about properties on page 1871.
Additional properties	The ServiceNow properties set by this task. Each property must be on its own line. For example, to update the state of the change request, you can specify or `state=-1`.

Table 1192. Wait actions

Wait action	Description
System ID	The ID of the ServiceNow change request. If you capture the ID with a create action **Output property**

Wait action	Description
	field, you can use it here. For example, if you specify `` `service_now_stage_ticket_id` `` in the **Output property** field of a create action, you can use the ID by specifying `` `${service_now_stage_ticket_id}` ``.
Field	The name of the affected field in the ServiceNow change request. The value in the field is checked for updates by the interval defined by the Check request interval field. For example, if you are checking the **approval** field, you might specify `` `approval` ``.
Value	The value expected in the ServiceNow field as defined by the **Field** property. For example, if you are checking the **approval** field, you might specify `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` ``, the task completes successfully.
Check request interval	The value defines how frequently ServiceNow is checked for updates. The value represents minutes. The default value is `1`, which mean ServiceNow is polled every minute.

Result

When the task starts, it has the status of **In progress**. The status changes to **Complete** when the related ServiceNow action is resolved.

What to do next

The ServiceNow change request life cycle is referred to as the state model. As a change request is used, it progresses through the various states of model. The state of a closed request is 'closed', for example.

In Measure, ServiceNow tasks are frequently used to change the state of a request. To modify a request state, you use the ServiceNow `state` property. For example, to change the state of a request to Review, you can specify **`state=0`** in the **Additional properties** field of an Update action.

The following table provides the values for the default states for the normal request type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 1193. ServiceNow normal request state model

State	Value
New	-5
Assess	-4
Authorize	-3

Table 1193. ServiceNow normal request state model (continued)

State	Value
Scheduled	-2
Implement	-1
Review	0
Closed	3
Canceled	4

Creating ServiceNow task using Service Now plug-in

A ServiceNow™ task can create and update ServiceNow change requests and the change tasks within the change request. You can also create a ServiceNow - Wait Change Request task which will wait for the specified value to appear in a specific field of a change request in ServiceNow.

Before you begin

You must have completed the following tasks:

- Ensured that you have internet-accessible instances of ServiceNow, such as `my_instance.service-now.com`.
- Created an integration with ServiceNow plug-in. For more information, refer to ServiceNow plug-in documentation.

Note: Ensure that the credentials used for the plug-in integration must have access to the ServiceNow API and permission to see and modify all change request fields. At a minimum, the user must be in a ServiceNow role that has the Create Change permission.

About this task

You can create the following automation tasks with which you can perform the actions in ServiceNow directly from Measure:

- **ServiceNow - Create Change Request and Change Task**

  This task creates a change request, change task or both in ServiceNow.

- **ServiceNow - Update Change Request and Change Task**

  This task updates the value in any change request, change task, or both in ServiceNow.

- **ServiceNow - Wait Change Request**

  This task waits for specified value to appear in a specified field of a change request in ServiceNow. The task will be completed when the provided value appears in the ServiceNow change request. The field can be any field of a change request.

- **ServiceNow - Wait Change Task**

  This task waits for specified value to appear in a specified field of a change task in ServiceNow. The task will be completed when the provided value appears in the ServiceNow change task. The field can be any field of a change task.

1. Click **Releases** on the left navigation pane.
   Result
   The **Releases** page is displayed.
2. Click any deployment plan.
   Result
   The **Deployment Plan** details page is displayed.
3. Click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
   Result
   The **Create Task** dialog is displayed.
4. In the **Type** list, select any one of the following options based on the action that you want to perform in ServiceNow:
   - **ServiceNow - Create Change Request and Change Task**
   - **ServiceNow - Update Change Request and Change Task**
   - **ServiceNow - Wait Change Request**
   - **ServiceNow - Wait Change Task**
5. In the **Name** field, enter a name for the task.
6. In the **Description** field, enter the description for the task.
7. In the **Integration** field, select the integration which you have created for ServiceNow.
8. Based on the task type you selected, enter the details in the task specific parameters, as described in the following tables:

Table 1194. ServiceNow - Create Change Request and Change Task

ServiceNow - Create Change Request and Change Task type	Description
Short description for change request	Enter the text that you want to display in the Short description field of the ServiceNow change request.
Request type	Enter the ServiceNow request type. you can use the ServiceNow default types or a user-created type. Default types are normal, emergency, and standard. Normal requests pass through all stages of the ServiceNow state model. Standard types are pre-authorized and skip the **Assess** and **Authorize** states. Emergency types skip the **Assess** state.

ServiceNow - Create Change Request and Change Task type	Description
Assignment group for change request	Enter the ServiceNow agent or group responsible for resolving the change request. Possible assignees include agents and groups available to the ServiceNow instance.
Additional properties for change request	Enter the ServiceNow properties that you want to update by this task while creating the change request. For example, a simple create change request might include the following fragments: {"approval:requested"}.
Create change task	Enter the ServiceNow properties that you want to update by this task while creating the change task within the change request. You can create multiple change task by separating each change task properties object by a comma. For example, a simple create change task might include the following fragments:[{"short_description":"createtask","change_task_type":"planning","description":"changetask"}]
Output property	Enter the system ID of the created change request.

Table 1195. ServiceNow - Update Change Request and Change Task

ServiceNow - Update Change Request and Change Task type	Description
Change Request Number	Select the Change Request Number of the ServiceNow change request that you want to update. See the Properties reference for information about properties on page 1871.
Change request properties	Enter the ServiceNow properties that you want to update by this task while updating the change request. For example, to update the state of the change request, you can specify or `{"state:-1"}`.
Update change task	Enter the ServiceNow properties that you want to update by this task while updating the change task within the change request. You can update multiple change task by separating each change task properties object by a comma. For example, to update the state of the change task, you can specify or `[{"state:3"}]`.

Table 1196. ServiceNow - Wait Change Request

ServiceNow - Wait Change Request type	Description
Change Request Number	Select the Change Request Number of the ServiceNow change request. See the Properties reference for information about properties on page 1871.
Field	Enter the name of the field for which the task should check for a match in the ServiceNow change request. The value in the field is checked for updates in a regular interval. For example, if you want to check for a match in the **approval** field, you might specify `` `approval` ``.
Value	Enter the value expected in the ServiceNow field defined in the **Field** property. For example, if you are checking the **approval** field, you can specify a value `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` `` in ServiceNow, the task completes successfully.

Table 1197. ServiceNow - Wait Change Task

ServiceNow - Wait Change task type	Description
Change task system id from property reference.	Enter the system ID of the change task.
Change Request Number	Select the Change Request Number of the ServiceNow change request. See the Properties reference for information about properties on page 1871.
Change request system id from property reference.	Enter the system ID of the change request.
Field	Enter the name of the field for which the task should check for a match in the ServiceNow change request. The value in the field is checked for updates in a regular interval. For example, if you want to check for a match in the **approval** field, you might specify `` `approval` ``.
Value	Enter the value expected in the ServiceNow field defined in the **Field** property. For example, if you are checking the **approval** field, you can specify a value `` `approved` ``. In this example, when the value in the **approval** field changes to `` `approved` `` in ServiceNow, the task completes successfully.

9. Click **Save** or **Save and create another** to add the task or add the task and create another task respectively.

Result

When the task starts, it will have the status of **In progress**. The status changes to **Complete** when the action mentioned in the task is performed in ServiceNow.

Example

Additional Information

The ServiceNow change request life cycle is referred to as the state model. As a change request is used, it progresses through the various states of model. For example, the state of a closed request is 'closed'.

In Measure, ServiceNow tasks are frequently used to change the state of a change request and change task.

To modify a request state, you use the ServiceNow `state` property. For example, to change the state of a change request to **Review**, you can specify "state:0" in the **Change request properties field** of a **ServiceNow - Update Change Request and Change Task** type and to change the state of a change task to **In progress**, you can specify "state:2" in the **Update change task** field of a **ServiceNow - Update Change Request and Change Task** type.

The following table provides the values for the default states for the normal change request type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 1198. ServiceNow normal change request state model

State	Value
New	-5
Assess	-4
Authorize	-3
Scheduled	-2
Implement	-1
Review	0
Closed	3
Canceled	4

The following table provides the values for the default states for the normal change task type. For more information, see the configure dictionary for the state property in the ServiceNow documentation.

Table 1199. ServiceNow normal change task state model

State	Value
pending	-5
open	1
in progress	2
closed	3
canceled	4

Creating DevOps Deploy tasks

Use DevOps Deploy tasks to run DevOps Deploy applications.

Before you begin

Before you can use DevOps Deploy tasks in pipelines and deployment plans, you must first configure DevOps Deploy integration on page 1885.

The applications that are available to a deployment plan depend on the team that is assigned to the plan. The applications that are managed by the team in DevOps Deploy are also available in Measure.

About this task

To create this task type, specify the application's process, version, and environment. You can set the version and environment at design time or select them at run time. When you run this task, the application runs in the integrated DevOps Deploy instance. During deployments, DevOps Deploy tasks start automatically when they become eligible to run.

To create a DevOps Deploy task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **DevOps Deploy**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, provide the description of the task.
5. In the **Integration** drop-down list, select the plugin integration.
6. Select the **Assigned group or user** from the drop-down list.
7. In the **Application** drop-down list, select an application.
8. In the **Process** drop-down list, select or enter an application process.

Processes that belong to the selected DevOps Deploy application are available.

9. In the **Version** drop-down list, select an application version.
   Versions refer to DevOps Deploy application snapshots. Versions that belong to the selected application are available. To postpone selecting a version, select **Use Version Tab**. If the application process does not require a version, select **No version items exist**. You might select this last option if you are running a configuration-type process that does not require components.
10. In the **Environment** drop-down list, select or enter an application environment.
    Environments that belong to the selected DevOps Deploy application are available. To postpone selecting an environment until you are ready to run the deployment, click **Versions** on the **Deployment plan** details page.
11. In the **Deploy Request Properties** box, enter the properties that you want to pass to the IBM DevOps Deploy (Legacy) application.
    Properties are name-value pairs. You can enter multiple properties. Put each property on a separate line. The following graphic provides an example of property definitions:
    Figure 30. property definitions

12. Select the snapshot display from the **On Failure** and **On Success** drop-down list.
13. Click **Save**.
    Result
    The task is inserted into the deployment plan.

Results

After the task is created, the plan's **Versions** page is updated with information about the application. If you selected **Only Changed Versions** for the application environment and version on the **Create Task** dialog box, use the **Versions** page to set those options before running the deployment.

Creating Ansible tasks

Create an Ansible® task.

Before you begin

In your Ansible® instance, create a user token. You use the token to authenticate your Measure user to Ansible®.

About this task

Create an Ansible® task to manage Ansible® jobs.

To create an Ansible® task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Ansible**.
3. In the **Name** field, enter a name for the task.
4. In the **Description** field, enter a name for the task.
5. In the **Instance URL** field, enter the location of your Ansible® installation..
6. In the **Access Token** field, paste the token that you generated in your Ansible® instance.
7. In the **Job ID** field, enter the Ansible® job identification number..
8. **Optional:** In the **Extra Variables** box, enter any parameters that you want to pass to the Ansible® job.
   The parameter format is JSON. For example:

   ```
   {
    "environment": "DEV"
   }
   ```

   example,
9. Click **Save**.
   Result
   The task is inserted into the deployment plan.

What to do next

When you run the task, a webhook is created that uses the Ansible® API to run the job. While the job is running, you can accesses Ansible® logs by using a link in the deployment plan. When the Ansible® job is finished, the Measure task status is updated.

Creating wait for approval tasks

A Wait for approval task pauses a deployment until a designated user provides approval. Email notifications are sent to users when the task starts.

Before you begin

Before using email tasks in your deployments, setup an email server. on page 1800

About this task

To create a wait for approval task, complete the following steps:

1. On the **Deployment Plan** details page, click **Create Task**.
   If you want to insert a task at a specific position in the plan, select a task before using the **Create Task**. The new task is inserted above the selected task.
2. On the **Create Task** dialog box, in the **Type** list, select **Wait for approval**.
3. In the **Name** field, enter a name for the task.
4. In the **Approval prompt** field, enter the text that you want to include in the email message.

The text is displayed in the **Task Details** section of the email.

5. In the **Approvers** list, select the users or groups designated to approve the task.
   Team members and groups are available to choose.
6. In the **Required approver** area, specify whether all the approvers must approve the task, or a single approver is sufficient.
7. Click **Save**.
   The task is inserted into the deployment plan.

Results

When the task runs, email notifications are sent to the selected approvers. The deployment plan cannot progress until the task is approved or skipped. Approval is granted by completing the task in the deployment plan.

Note: For information on pipeline notifications, see this page on page 3149.

Approving wait for approval task

You can approve the wait for approval task using the received email notifications.

About this task

Designated approval user receives the email notification for approving the wait for approval tasks. The email notification contains the task information for which the wait for approval task needs to be approved.

You can click **Approve** or **Reject** for approving or rejecting the wait for approval task.

By default, designated users can approve the task or share the email notification to any user for approving. You can prevent the latter functionality by enabling the `authorized-approval` feature flag.

To activate this security feature, enter the feature name `authorized-approval` under `featureFlags` . After enabling `authorized-approval` feature flag , only designated users can approve or reject the wait for the approval task. To approve or reject, the user must login Measure. Not designated users are not authorized to approve or reject the wait for approval task using the email notification.

Using properties

A deployment plan can contain user-defined properties. Properties values can be defined at design- or run-time.

Tasks can reference properties defined in their parent deployment plan. In addition to the properties that you create, system properties are available to all plans.

Creating properties

Creating properties

About this task

You define properties with the **Deployment Plan** details page by clicking **Properties**. You can create a property before you use it, or you can create a property by first referencing it in a task, and then later defining its attributes. If you create a property by reference, it appears as text-type property on the **Properties** tab.

To create a property, complete the following steps:

1. On the **Deployment Plan** details page, click **Properties**, and then click **Add**.
2. On the **Add property** window, in the **Name** field, enter the name. Property names cannot contain white spaces, `{}`, or `:`.
3. In the **Label** field, enter the label that appears in the user interface.
4. In the **Type** list, select the property type.
   The available types are described in this list:
   - `Text`
   - `Textarea`
   - `Password`
   - `Select`
   - `Boolean`
   - `Date and time`. Values are replaced by a string with the following format `day, month, year, time`. For example, `Thursday, January 1st 1970, 12:00 am UTC`.

     Note: When a date and time property is used in a ServiceNow® task, the value is replaced with the date and time format expected by ServiceNow®.

5. In the **Value** field, you can enter an initial value for the property.
6. In the **Default Value** field, enter a value for the property.
   This value is used unless the user sets the value. You can leave this field blank.
7. If you want users to supply a value for the property when they create a release, check **Prompt for this property at release creation at the specified sequence position**.
8. In the **Display sequence position** list, select a position for the property.
   This selected value affects the property's position on the **Properties** tab
9. Click **Save**.
   Result
   The property is displayed on the deployment plan **Properties** page.

What to do next

After the release is created, you can add deployment plans, events, and tags to it.

Using properties

You can use properties in deployment plans, REST API endpoints, and external systems, such as ServiceNow®.

Property examples

Use the following syntax to reference a property: `` `${propertyName}` ``. To escape the syntax pattern, use two dollar-sign symbols, `$${...}`. You set property values with the deployment plan's **Properties** page, REST API endpoints, or external systems, such as ServiceNow®.

The following table describes how to properties with task types.

Task type	Usage
ServiceNow	You can use the ServiceNow on page 1863 **Output property** field in a create change request action to store the change request ID in a property, for example, `'service_now_prod_ticket_id'`. Then, with another task, update the change request by referencing the property in the **System ID** field like this, `'${service_now_prod_ticket_id}'`.
Email	You can reference a system property in the email task on page 1856 **Subject** field, for example, `'${sys:release.name}' is complete!`.
Delayed	You can reference a delay task on page 1856 property in the **Delay property** field, for example, `'${prodTarget}'`. In this example, the task delays until the value in the `'${prodTarget}'` property is reached.

System properties

System properties enable you to access information about the parent release of a deployment plan. System properties can be referenced by any property-enabled task that is part of a release.

The following table contains the available system properties:

Table 1200.

Property	Description
`${sys:release.name}`	The value in the release's **Name** field.
`${sys:release.description}`	Contains the content of the release's **Description** field.
`${sys:release.start}`	Release's **Start time**.
`$${sys:release.end}`	Contains the content of the release's **End time** field.

In the **Email message** field of an email-type task, you can reference system properties like this example:

```
Deployment for release `${sys:release.name}` - `${sys:release.description}` is complete.
```

You can reference a system property in a ServiceNow task's **Short description** field like this example:

`Staging deployment `${sys:release.name}``.

Extending Product Functions

Learn how to extend the function of Velocity Measure

Measure is designed to be easily extended. You can enhance existing functions or add entirely new ones by configuring plugin integrations. Plugins are software add-ons that extend Measure functionality by integrating with external tools. In general, each plugin manages one external application. The HCL DevOps Deploy (Deploy) plugin, for example, enables you to run Deploy applications.

If you cannot find a plugin for your environment, you can create plugins by using the plugin software development kit (SDK) on page 1884.

Measure supports several plugin types:

- Value stream plugins are supported by default. To create an integration with these plugins, you simply configure an integration in the value stream JSON file.
- Quality data plugins integrate test and metrics data into reports and value streams on page 1893. To create an integration with these plugins, you use external clients to post data to Measure API endpoints.
- Dedicated plugins provide external data to value streams, pipelines, and releases. Plugins of this type include: Deploy on page 1885, and Jenkins on page 1886. Use the Integrations page to configure integrations for these plugins.
- Task-based integrations provide external data to deployment plans. Use ServiceNow tasks to manage ServiceNow change requests on page 1887.

Introducing plugins

Extend Measure by configuring plugin integrations to external tools.

Measure uses a containerized microservices architecture. This means that plugins are not run directly from Measure but from containerized instances managed by a containerization platform. Runtime instances are created from images on the Docker Hub™ repository. Measure retrieves images from this repository and creates instances as needed.

For Kubernetes-based installations such as OpenShift™, Measure uses Kubernetes™ as its containerization platform. An Argo Workflow Engine manages plugin workflow. A Kubernetes pod starts when a plugin task is requested and sends the results to Measure. When complete, the container containing the plugin ends.

Plugins are categorized by data collection and communication methods. Generally, plugins are designed to use one of the following communication methods.

- **Webhook**. These plugins use webhooks to communicate with a defined Measure API endpoint. The webhook is used to trigger data collection events. Examples include the AppScan™, and SonarQube plugins.
- **Poller**. These plugins are based on an event defined by the plugin. Queries are performed to determine when to send and update data from the external service. Examples include the GitHub™, and Rally plugins.
- **Parser**. These plugins integrate functions to parse a specific file type and create a metric document that Measure can display with other value stream management and portfolio views. Examples include the JUnit, and OneTest plugins.

Available plugins are stored at HCL plugin database.

If you cannot find a plugin for your environment, you can create plugins by using the plugin software development kit (SDK) on page 1884.

If a metrics plugin is not available for your tool, you can upload custom metrics using Measure API endpoints.

To use a plugin, you configure an integration. "Integration" refers to a user-configured instance. You might use the GitHub plugin to configure an integration with the Service A repository, and then configure another integration with your Service B repository. Although both integrations use the GitHub plugin, each integration provides a unique set of data to Measure. You manage each integration separately.

Some integrations are termed "native integrations," and are not technically plugin derived. Native integrations are used with deployment plans by defining tasks of a specific type on page 1885. You can use DevOps Deploy data with your deployments and pipelines, for example, by defining DevOps Deploy-type tasks.

Some plugins might have a corresponding plugin in the external tool. To integrate Jenkins, for example, you install the " Measure" plugin into Jenkins™ using the Jenkins plugin manager. Then you configure a corresponding Jenkins integration in Measures. The two integrations operate in tandem.

Preventing plugins installs in unsupported server versions

Plugin versions incompatible with Measure are prevented from installing.

About this task

Based on the your installed version of Measure, plugins not compatible with that version are prevented from installing in the product. If the installed version of Measure is equal or greater than the product version that the plugin supports, the installation of the plugin is possible.

Notifications for incompatible plugins versions prevented from installation are displayed in each of the following three scenarios:

Scenario 1: Installing plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Plugins** tab.

3. Locate the required plugin in the list and click **Install**.
4. The error **Plugin <name> is only supported on** Velocity **versions equal to or greater than x.x.x** is displayed on the right side of the **Status** field for that plugin.

Scenario 2: Loading plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Plugins** tab.
3. Click **Load plugin**.
4. In the **Load a custom plugin** window, click in the **Plugin Image** field and enter the required plugin.
5. Click **Submit**.
6. The error **Plugin <name> is only supported on** Velocity **versions equal to or greater than x.x.x** is displayed in the **Load a custom plugin** window.

Scenario 3: Upgrading plugin

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click the **Integrations** tab.
3. Locate the required plugin and then click on the overflow menu and select **Upgrade**.
4. In the **Status** field for the plugin, **Upgrade failed** is displayed.

Creating plugin integrations

Defining integration allows you to communicate with external tools.

Before you can create an integration, ensure you have the user credentials and connection information required by the external tool, such as URLs, and API endpoints. To create an integration with HCL DevOps Deploy (Deploy), for example, you need a user token generated by Deploy, and the URL where the server is located. Ensure that you have all the necessary connection information before starting.

Credentials and other connection properties vary among individual plugins. You can find property descriptions and use cases for all plugins on the product website.

A plugin is a software add-on designed to communicate with external services and applications. Each plugin defines an expected record type and communication method. Communication can be uni-directional or bi-directional. To use a plugin, you must install and configure an integration.

To view the plugin in Measure, on the **Home** page, click **Settings > Integrations**.

The **Integrations** page has three tabs:

1. Integrations on page 3249
2. Installed Plugins on page 3249
3. Available Plugins on page 3250

Integrations

The **Integrations** tab lists all plugins that are configured for integration. Click the right-angle bracket to view the details of the configured plugin. All integrations are listed regardless of how they are created.

You can use search to find configured plugin integrations. The search query finds matches in the **Name** and **Type** fields. On the **Integrations** tab, you can sort by the **Name**, **Status**, **Type**, and **Last sync** fields in ascending or descending order by clicking a table header.

Click **Show Archived** to see the archived plugin integrations. The following table details the fields on the **Integrations** tab:

Field	Description
Name	The name of the plugin. Click the right-angle bracket to view details of the plugin integration. Click to view plugin documentation.
Status	The working status of the configured plugin integration. Possible values are **Online** and **Failed**. The **Online** status is displayed for working plugin integrations. The **Failed** status is displayed for plugin integrations that are not working. Disrupted communications or inaccurate integration configurations can cause the integration to fail.
Type	The type of the plugin.
Last sync	The last plugin synchronization time.
Disable/Enable	Set to disable or enable the plugin integration.
	Click to **Edit**, **Upgrade/Downgrade**, **View logs**, and **Archive/Delete** plugin integrations.

Installed Plugins

The **Installed Plugins** tab displays all plugins that are installed by default with the Measure installation. The installed plugins are ready to configure for integration.

You can search for installed plugins. The search query finds the matches in the **Name** field. On the **Installed Plugins** tab, you can sort the **Name** field in ascending or descending order by clicking the header.

The following table details the fields on the **Installed Plugins** tab:

Field	Description
Name	The name of the plugin. Click to view all installed versions of the plugin. Click to view plugin documentation.
Version	The version of the installed plugin.
Image	The image of the plugin.
Release Date	The release date of the installed plugin.
Actions	Click **Add integration** and provide details to configure the integration. All the integrations are enabled by default. To create a disabled plugin integration perform the following the steps: • In the **Add integration** window, click **Show hidden properties**, and then clear **Enabled** to create a disabled plugin integration. Click and select **Delete** to delete the installed plugin.

The latest version for Parser plugins are installed with the **Non-Configurable** status.

Available Plugins

The **Available Plugins** tab displays all supported plugins for Measure.

You can search for the supported plugins. The search query finds matches in the **Name** field. On the **Available Plugins** tab, you can sort the **Name** field in ascending or descending order by clicking on the header.

The following table details the fields on the **Available Plugins** tab:

Field	Description
Name	The name of the plugin. Click to view all versions of the plugin. Click to view plugin documentation.
Version	The version of the available plugin.
Image	The image of the plugin.
Release Date	The release date of the available plugin.

Field	Description
Actions	Click **Install** to install the required version of the plugin.

Advanced usage

You can configure integrations for some plugin types by adding an integration definition to a value stream JSON file on page 3078. Using a text editor, define an integration object in the `integrations` array of the `value_stream_name-vsm.json` file. The following code example illustrates a Jira integration.

```
{
     "_id": "velocity-generated-id",
     "type": "ucv-ext-jira",
     "tenant_id": "velocity-generated-id",
     "name": "my-JIRA",
     "disabled": false,
     "properties": {
       "baseUrl": "http://my-jira-url",
       "username": "admin",
       "password": "****",
       "jiraProjects": [
         "My-Project"
       ]
     }
   },
```

Integrations created by the value stream method are displayed on the **Integrations** page along with integrations created by the other methods.

Additional Measure product features requiring you to perform plugin installation before use include the following:

1. Handlebars Reporter
2. Bottleneck detection
3. Parser

The integration is listed on the **Integrations** page. The **Integrations** page is where you manage integrations. on page 1881

Integrations start providing data when successfully configured. For some integration types, such as quality-data types on page 3165, you can start using their data without additional configuration. For example, you can create dashboard charts that contain quality data when the integration is added. For example, JUnit is a quality-type plugin.

Other integration types require additional configuration before you can start using the associated data. Native integrations require that tasks are added to deployment plans. on page 1885 Value stream integrations, such as GitHub™, require a reference in the value stream JSON file. If you define an integration on the **Plugins** tab, add a reference to the integration name in the JSON file. If you define an integration directly in a JSON file described in Step 3, no additional configuration is required.

Github plugin integration example

Github plugin integration example

About this task

To configure the Github plugin integration, complete the following steps:

1. In the **Integration Name** field, enter a name for the integration, such as "Git tutorial."
   Such as GitHub tutorial. Later, we use this name to associate the integration to our value stream.
2. In the **Url** field, enter the URL for the GitHub project that you configured earlier .
   The URL is similar to this example, `https://github.com/myGitHubID/myGitHubProject`.
3. In the **Name** field, enter the name of the GitHub repository.
   Using the example URL from the preceding step, the name is `myGitHubProject`.
4. In the **Owner** field, enter the GitHub account that owns the repository.
   Using the example URL, the owner is `myGitHubID`.
5. In the **Token** field, enter your Github personal access token to authenticate with the Github repository.
   In GitHub, to generate a personal access token, go to **Settings**>**Developer settings**>**Personal access tokens** then click **Generate new token** and select the scope **repo** which allows to access repositories.
6. In the **API URL** field, enter the GitHub API URL.
   For enterprise Github account, use GitHub API URL `https://example.com/api/v3`.
   For public Github account, use GitHub API URL `https://api.github.com`.
7. Select the **Show Hidden Properties** check box.
   Result
   The hidden fields are displayed.
8. Enter the custom time for the plugin job in the **Plugin Job Timeout** field.

   Note: The default time is 1h, you can specify the maximum time a plugin job should be allowed to run before being automatically canceled in the following format:
   - 30m for 30 minutes
   - 1h for one hour
   - 1d for one day

9. **Optional:** Select the **Interval** time to set the frequency to run the schedule event.

   Note: If you do not want to change the interval time of the schedule event while creating the integration, the default sync intervals defined in the plugin are applied.

10. Click **Add**.

Results

The integration is listed on the **Integrations** page. If the integration is successful, the **Status** field displays `Online`. If the status is not `Online`, use the **View log** option to troubleshoot the integration. You can increase the logging level in the integration configuration if you need more troubleshooting information.

Managing integrations

You can manage the existing configured integrations.

About this task

After an integration is successfully configured, it periodically communicates with the external tool to synchronize the data flow.

The default communication interval when scheduled event integrations run can be configured by the `PLUGIN_INTEGRATION_RUN_INTERVAL` environment variable on the reporting-consumer service to a valid integer for the minutes between each run.

To manage an integration, perform the following steps:

1. On the Measure **Home** page, click **Settings > Integrations**.
   Result

   The **Integrations** page lists all configured integrations. All integrations are listed regardless of how they are created. You can use search functionality for searching the integration in the **Integrations** tab. The search query finds the matches in **Name** and **Type** fields. On the **Integrations** tab, you can sort the **Name**, **Status**, **Type**, and **Last sync** fields in ascending or descending order by clicking on the required table header field.
2. **Optional:** To modify an integration definition, click overflow menu ⋮ **> Edit**, and then modify the integration definition properties.
3. **Optional:** To view the integration log file, click overflow menu ⋮ **> View logs**.
   This action displays the integration's last execution log.
4. **Optional:** To upgrade or downgrade the plugin used by the integration, click overflow menu ⋮ **> Upgrade/ Downgrade**.
5. **Optional:** To sync with the external tool, click overflow menu ⋮ **> Resync**.
   This action attempts to communicate with the external tool immediately. You might do this when you want to run tests before the next communication interval. You can also select the time line from when you want to resync the data from the external tool.

   Note:

Not all integration types support this action. All scheduled event plugins such as Jira, GitHub, GitLab, Azure DevOps, and HCL DevOps Deploy are supported with the resync functionality.

6. **Optional:** To archive an integration, click overflow menu **> Archive/Delete**.

   Note: Select the **Delete the integration data** checkbox to delete the integration data.

7. **Optional:** To disable or enable an integration, click the **Disable/Enable** field of the integration.
   Result
   Communication with the external tool stops until re-enabled. You might do this if you want to stop data delivery during testing.
8. **Optional:** To disable or enable multiple integrations, select the required integrations and then **Select Actions** appears. Click the drop-down list of **Select Actions** and select **Disable** or **Enable**.

Defining an initial sync date for plugins

For scheduled events plugins, you can specify an initial sync date.

About this task

For scheduled events plugins, you can specify an initial sync date by performing the following steps:

1. On the **Settings** page, click **Integrations**.
2. In the **Integrations** page, click **Plugins** tab.
3. Locate the required plugin and then click **Add integration**.
4. In the **Add Integration** window, click the **Show hidden properties** check box.
5. Click **Initial Sync Date** field to specify the date to pull data for initial run.
6. Fill the remaining required fields in the **Add Integration** window.
7. Click **Add**.

Note: If the plugin does not support the sync feature, the **Initial Sync Date** field is not displayed when you click the **Show hidden properties** check box in the **Add Integration** window.

Archiving integrations

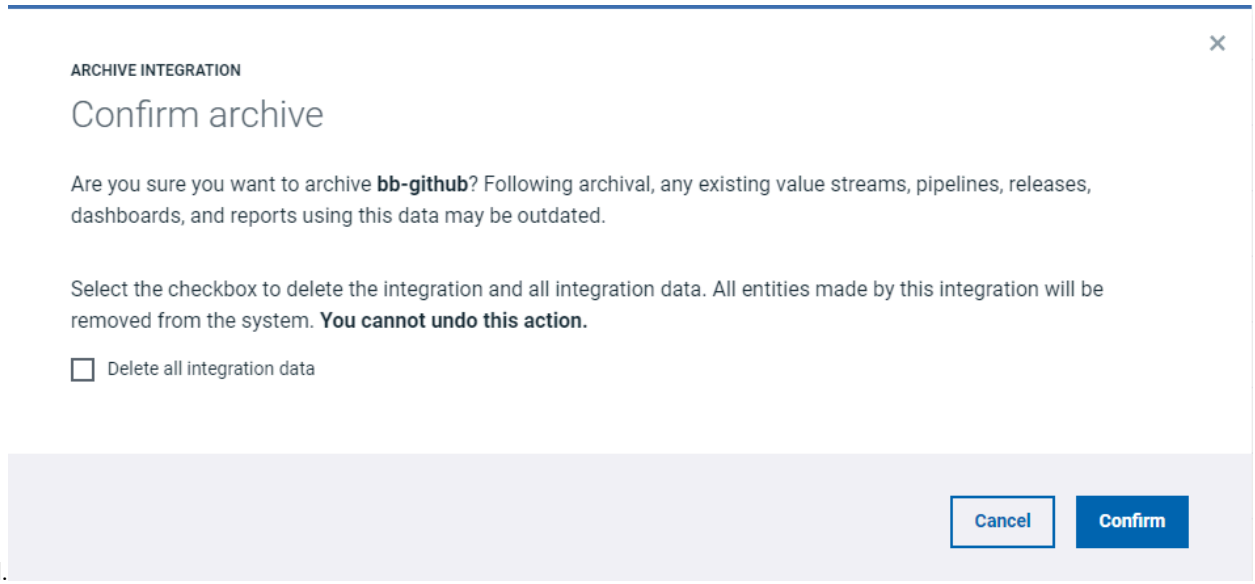
Archive the no longer needed integrations.

About this task

This topic describes the procedure for archiving integrations.

To archive integration, complete the following steps:

1. On the **Home** page, click **Settings > Integrations > Configured**.
   Result
2. Select **Archive/Delete** from the overflow menu (⋮) of the plugin integration which you want to archive.
3. The archive confirmation modal window is displayed.



4. Select **Delete all integration data** checkbox to delete all integration data and click **Confirm**.
   You can view all archived integrations by clicking **Show Archived**.

Note: You can restore the archived integrations by clicking the overflow menu and select the **Restore**.

Upgrading and downgrading integrations

You can upgrade or downgrade the integration versions based on the plugin's compatibility with Measure.

Before you begin

You must have installed the required plugin. See Creating plugin integrations on page 1876.

1. On the **Home** page, click **Settings > Integrations**.
   The **Plugin and Integrations** page is displayed.
2. Click **Integrations**.
3. Select the toggle button under the **Disable/Enable** field to disable the plugin integration for which you want to upgrade or downgrade the version.
   You can avoid a partial data synchronization issue by disabling the integration before the upgrade.
4. Select **Upgrade/Downgrade** from the overflow menu (⋮) .

Note: The plugin version must be compatible with the Velocity version. For more details about the compatibility matrix, see the specific plugin documentation.

5. Select the version from the drop-down list.
6. Click **Upgrade**.

Note: If you select a lower version of the plugin from the drop-down list, the integration downgrades to the lower version.

7. Select the toggle button under the **Disable/Enable** field to enable the plugin integration.

Results

You have upgraded or downgraded the integrations.

Creating plugins with the plugins SDK

You can create your own plugins and install them in Measure. After installing custom plugins, you can integrate them in your releases, reports, and value streams.

Measure provides a SDK that contains the tools, including local dependencies and wrappers, that you need to create your own plugins. The plugin SDK provides a sample plugin that you can use to get started, and JavaScript templates that you can customize for your own plugins.

To create a plugin, your development environment must include the following items:

- Node.js version 12 or later
- NPM version 6 or later
- Docker
- Measure version 2.0 or later

To get started with the plugin SDK, complete the following steps:

1. Download the plugin SDK. On the Measure **Plugins** tab, click **Download plugin sample**. After the download is finished, extract the compressed file.
2. Build the sample plugin. In the `ucv-ext-sdk-template` folder, run the following commands. Additional instructions are provided in the SDK `Readme.md` file.

```
 npm install
npm run dist-runnable
docker build . -t name:tag
```

The `name` value can be any value that you want. The `tag` value must represent the latest image version, or some other valid semantic version. For example, `docker build . -t testplugin:0.1`.

3. Upload the sample plugin. On the Velocity **Plugins** tab, click **Load plugin**, and then select the Docker image that contains your plugin. To upload the image from the previous step, you specify `testplugin:0.1`.
4. Create an integration by using the sample plugin. On the **Plugins** tab, click **Add integration**.

Plugins are categorized by data collection and communication methods. Generally, plugins are designed to use one of the following communication methods.

- **Webhook**. These plugins use webhooks to communicate with a defined Measure API endpoint. The webhook is used to trigger data collection events. Examples include the AppScan™, and SonarCube plugins.
- **Poller**. These plugins are based on an event defined by the plugin. Queries are performed to determine when to send and update data from the external service. Examples include the GitHub™, and Rally plugins.
- **Parser**. These plugins integrate functions to parse a specific file type and create a metric document that Measure can display with other value stream management and portfolio views. Examples include the JUnit, and OneTest plugins.

The plugin SDK provides JavaScript templates for each type. For example, the template `ucv-ext-sdk-templates/src/endpoints/SampleEdnpoint.js` defines the properties that you need to create an endpoint-type plugin.

Managing native integrations

Native integrations add external functionality to releases, deployment plans, and pipelines.

Native integrations are available for the following applications or services:

- IBM DevOps Deploy
- Jenkins™
- ServiceNow™
- Ansible™

Configuring DevOps Deploy integrations

Use DevOps Deploy applications in your deployments and pipelines.

Before you begin

- Obtain a token from your DevOps Deploy server. When you create the DevOps Deploy token, specify **admin** as the user.

  This task requires that you are an Measure administrator.
- You must install the DevOps Deploy plugin, see Github plugin integration example on page 1880.

Note:

You must not integrate the same instance of Deploy using UrbanCode Deploy (Legacy) and Deploy (ucv-ext-ucd) plug-in in Measure. Integrating the same Deploy instance using UrbanCode Deploy (Legacy) and Deploy plug-in in Velocity can result in data corruption.

About this task

After you configure an integration, create tasks to run DevOps Deploy applications during your deployments on page 1868. You can also add applications to your pipeline stages and run them that way.

1. Click **Settings > Integrations**.
   The **Plugin and Integrations** page is displayed.
2. Click **Installed Plugins**, and click **Add Integration** for **DevOps Deploy**.
3. In the **Add DevOps Deploy Integration** window, enter a name for the integration, and in the **URL** field, specify the URL. The format is `HTTPS://hostname:port`. For example, `HTTPS://sampleUCD:8443`.

   Note: If you are running DevOps Deploy locally, you can substitute `HTTPS://host.docker.internal:port` in place of `localhost`.

4. In the **Access Token** field, paste the token from DevOps Deploy.
5. Click **Add**.

   Note: The initial synchronization includes four months of data. Depending on the amount of data, the synchronization process can take some time.

What to do next

Synchronization occurs every fifteen minutes but you can synchronize any time by clicking **Actions > Resync** on the **Integrations** page. To prevent synchronization, select the toggle button under **Disable/Enable** field on the **Integrations** page.

Configuring Jenkins integrations

Manage Jenkins™ jobs with your deployments and pipelines.

Before you begin

Obtain the Measure plugin and install it in your Jenkins instance. On the Jenkins Plugin page search for `velocity`. Complete the integration by pasting the Measure credentials into Jenkins.

Note: Ensure the following requirements are met prior to installation of the plugin.

- You must have admin privileges in Jenkins.
- Upgrade the Jenkins plugin version to 2.1.0 in order to upgrade the Velocity version to 2.1.0 and later. Velocity version 2.0.6 will be the last version to support the previous Jenkins plugin version of 2.0.6 and earlier.

Note:

You must not integrate the same instance of Jenkins using Jenkins (Legacy) and Jenkins (ucv-ext-jenkins) plug-in in Measure. Integrating the same Jenkins instance using Jenkins (Legacy) and Jenkins plug-in in Velocity can result in data corruption.

About this task

This configuration makes Jenkins jobs available to pipelines and deployment plans. You can run Jenkins jobs with Measure and use the output in later tasks. After you configure an integration, add Jenkins tasks to deployment plans on page 1858 and pipeline stages.

1. Click **Settings > Integrations > Plugins**.
   A list of defined integrations with current information about each integration is displayed in a table.
2. Click **Add Integration** for **Jenkins**.
3. In the **Add Jenkins Integration** window, in the **Name** field, enter a name for the integration and click **Create**.
4. In the **Integration ID** field, click to copy the ID and then paste it into the corresponding field in your Jenkins instance, **Manage Jenkins > Configure System > DevOps Velocity**.
5. In the **Integration token** field, click to copy the token and then paste it into the corresponding field in Jenkins.
6. Click **Save**.
   Depending on the amount of data,the initial integration can take some time.
   Result
   Synchronization occurs every minute. To prevent synchronization, click the **Disable/Enable** field on the **Integrations** page.

What to do next

After the initial integration, you can add Jenkins tasks to deployment plans and select any of the integrated jobs. Similarly, you can add Jenkins jobs to pipeline stages.

Integrating with ServiceNow

Create and manage ServiceNow™ change requests with Measure. Update change requests with Measure tasks and then react to status changes. Control deployment progress with change request updates.

ServiceNow integrations are managed on the task level. You configure an integration when you add a ServiceNow task to a deployment plan on page 1859. After an integration is configured, other ServiceNow tasks can use the same instance.

Migrating integrations

Migrate the legacy plug-in integrations to external plug-in integrations.

You can migrate the following (Legacy) plug-in integration to external plug-in integrations:

Migrating integration of Jenkins (Legacy) to Jenkins

Migrating integration of Jenkins (Legacy) to Jenkins.

Before you begin

You must have completed the following tasks:

- Back up the database, see instructions in this page.
- Ensure that you have added the version parameter to all applicable jobs in Jenkins server. The Jenkins plugin search for jobs with version parameter as reference.

About this task

To migrate the integrations of Jenkins (Legacy) to Jenkins, complete the following steps:

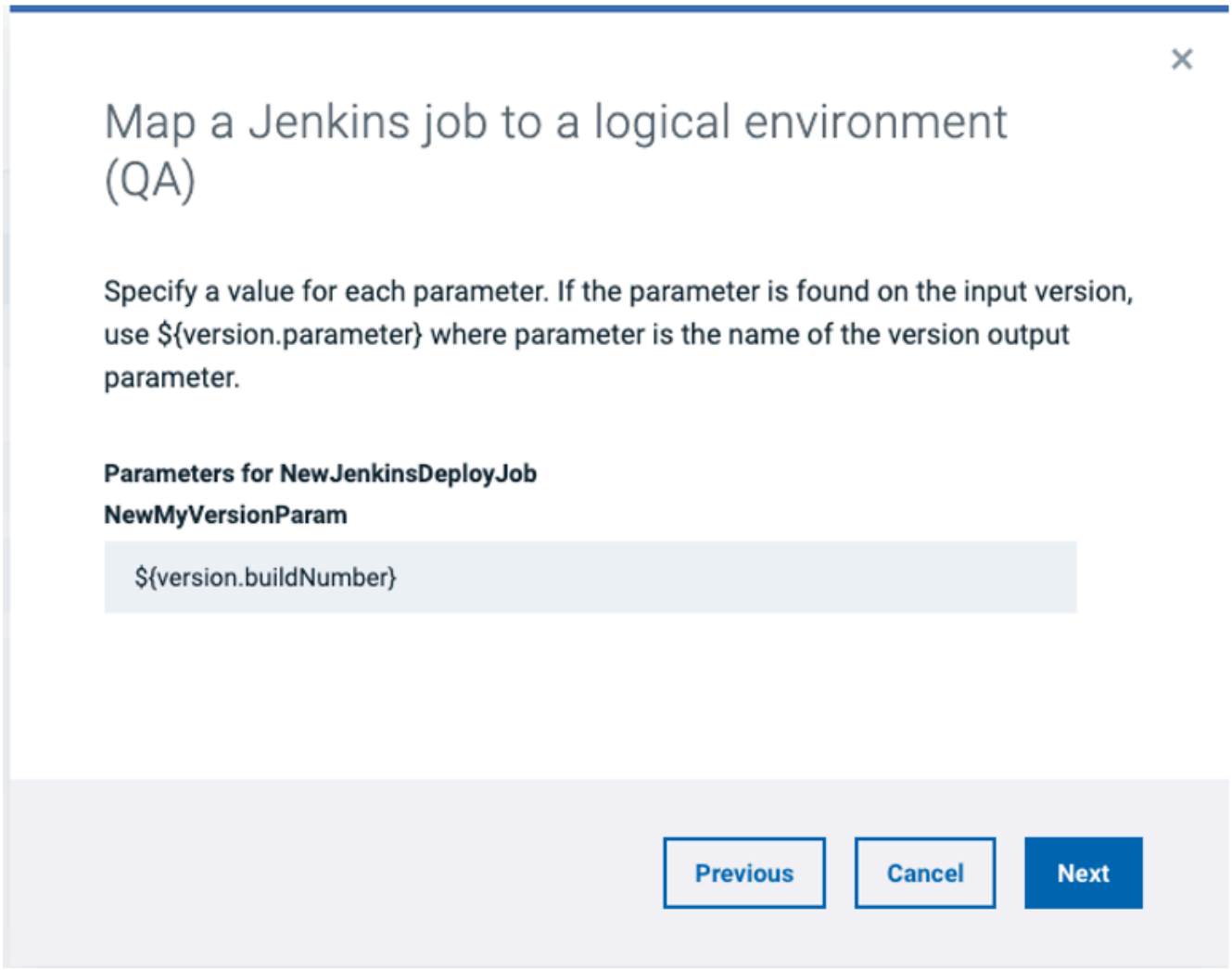
1. From the Measure page, click **Settings > Integrations > Configured**.
   A list of all configured integrations are displayed.
2. Click overflow menu for configured **Jenkins (Legacy)** plug-in and then click **Migrate**.
   The **MIGRATE JENKINS INTEGRATION** window is displayed. You can view the road-map of migration process. Before you begin the migration process, ensure that you have taken the backup of database.
3. On the **MIGRATE JENKINS INTEGRATION** window, click **Continue**.
   The **Add Jenkins Integration** window is displayed.
4. On the **Add Jenkins Integration** window, enter the value for all required fields to create new integration of **Jenkins** and then click **Add** to add the integration details.
   The migration process starts automatically and it reaches to **complete** step of migration process.
5. Click **Close** to close **Migrate Jenkins Integration window**.
   You have successfully migrated integration of **Jenkins (Legacy)** to **Jenkins**.

Results

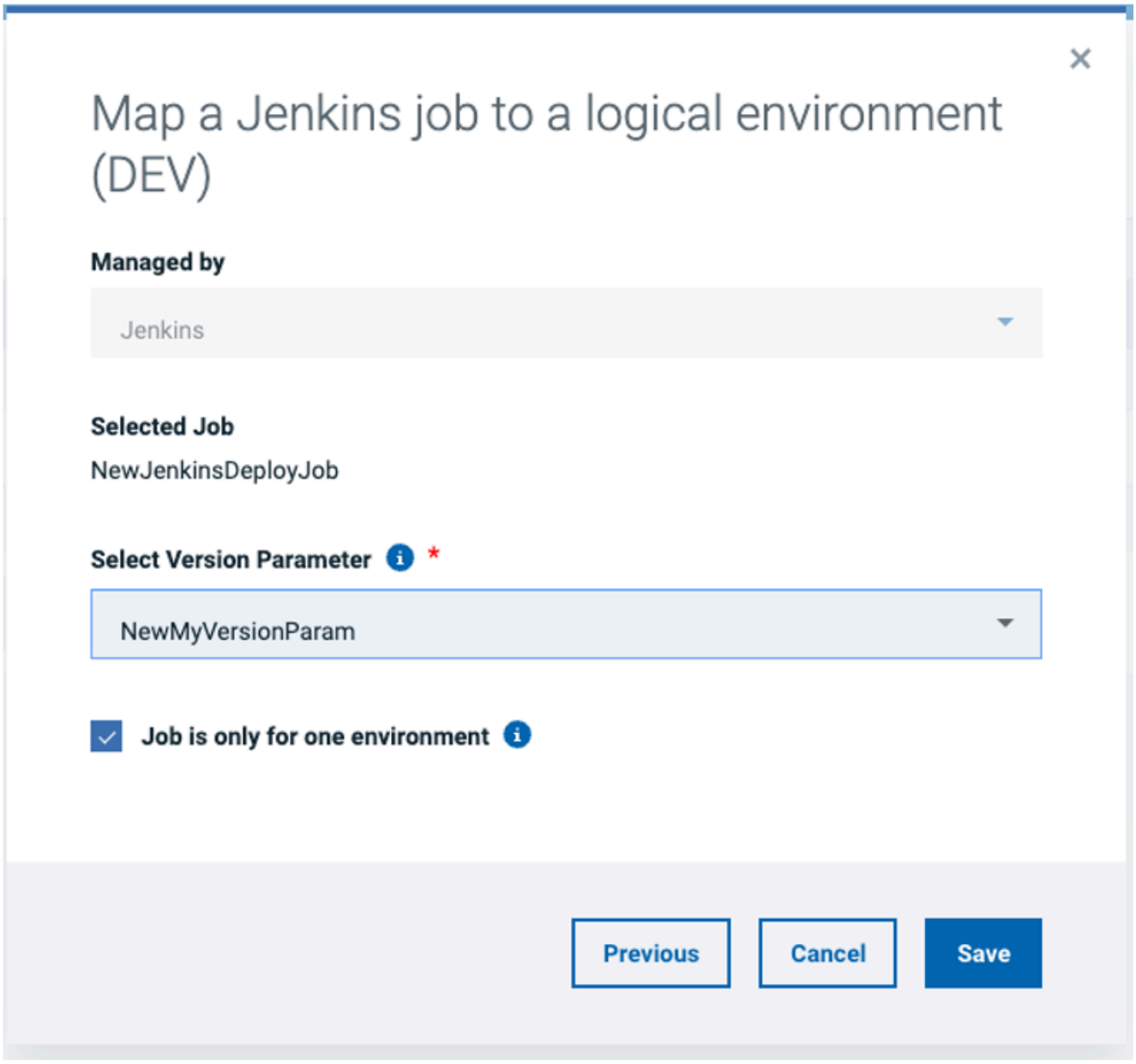
You can see the **Jenkins** integration data in Velocity.

What to do next

Go to pipeline of any value stream and add Jenkins application. While configuring the Jenkins job to logical environment ensure that you use `${version.buildNumber}` parameter.


Map a Jenkins job to a logical environment (QA)
Specify a value for each parameter. If the parameter is found on the input version, use ${version.parameter} where parameter is the name of the version output parameter.
Parameters for NewJenkinsDeployJob
NewMyVersionParam
${version.buildNumber}
Previous
Cancel
Next

Choose the required parameter in **Select Version Parameter**



field.

Migrating integration of UrbanCode Deploy (Legacy) to HCL DevOps Deploy (Deploy)

Migrating integration of UrbanCode Deploy (Legacy) to DevOps Deploy.

Before you begin

You must have completed the following tasks:

- Ensured that you have taken Back up the database, see instructions in this page.
- Ensured that there is no **Releases** with **UrbanCode Deploy (Legacy)** task is **In progress** or **Scheduled** for later.
- Ensured that there is no **UrbanCode Deploy (Legacy)** deployment is **In progress** or **Scheduled** for later.

About this task

To migrate the integrations of UrbanCode Deploy (Legacy) to DevOps Deploy, complete the following steps:

1. From the Measure page, click **Settings > Integrations > Configured**.
   A list of all configured integrations are displayed.
2. Click overflow menu for configured **UrbanCode Deploy (Legacy)** plug-in and then click **Migrate**.
   The **MIGRATE URBANCODE DEPLOY(HCL LAUNCH) INTEGRATION** window is displayed. You can view the stages of the migration process.
3. On the **MIGRATE URBANCODE DEPLOY(HCL LAUNCH) INTEGRATION** window, click **Continue**.
4. The **URBANCODE DEPLOY (Legacy)** integration is migrated to DevOps Deploy integration. The integration name is changed by appending **--migrated-EXT**.
   The migration process automatically installs latest available version of the DevOps Deploy plugin. If any version of DevOps Deploy plugin is already installed, the migration process uses that version of the plugin for migrating the integration.

Results

All **URBANCODE DEPLOY (Legacy)** task type changes to DevOps Deploy task in templates.

What to do next

You can schedule the deployment for DevOps Deploy applications by using the migrated DevOps Deploy integrations.

Uploading a build to a pipeline

builds/apiDriven

```
curl -k -X POST 'https://localhost/api/v1/builds/apiDriven' \
    -H 'Authorizat1on: UserAccessKey <User Access Key>' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":"<Version Name>",
        "application":{"externalId":"efecb73f-dclc-4d3d-b3bb-
        38339445e892"},
        "url":"<Build URL>",
        "revision":"<Commit SHA>"
    }'
```

Table 1201. builds/apiDriven parameters

Parameter	Type	Required	Description
UserAccesKey	string	false	User access key
version	string	true	JSON object with either 'externalId' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})

Table 1201. builds/apiDriven parameters (continued)

Parameter	Type	Required	Description
application	object	false	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
url	URL	false	URL link to build in external tool
revision	string	true	SHA of the commit linked to the build

Example

Example

```
curl -k -X POST 'https://localhost/api/v1/builds/apiDriven' \
    -H 'Authorizat1on: UserAccessKey 3b3ec6cc-1e37-4353-bc3a-46b493eed4e6' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":"1.0",
        "application":{"externalId":"b900ff5b-1fff-44ab-bcc6-9d39d5937026"},
        "url":"https://test.com",
        "revision":"12345678"
    }'
```

Uploading deployment status

deployments/apiDriven

```
curl -k -X POST 'https://localhost/api/v1/deployments/apiDriven' \
    -H 'Authorizat1on: UserAccessKey <User Access Key>' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":{"name":"<Version Name>",
        "application":{"externalId":"efecb73f-dclc-4d3d-b3bb-
        38339445e892"},
        "url":"<Deployment URL>",
        "environment":{"name":"DEV"}
    }'
```

Table 1202. deployments/apiDriven parameters

Parameter	Type	Required	Description
UserAccesKey	string	false	User access key

Table 1202. deployments/apiDriven parameters (continued)

Parameter	Type	Required	Description
version	object	true	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
application	object	true	JSON object with either 'externalid' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})
url	URL	false	URL link to build in external tool
environment	object	true	JSON object with either 'externalId' or 'name' (Examples: {"name": "apidrivenjenkinsjob"} or {"name": "application id in external tool"})

Example

Example

```
curl -k -X POST 'https://localhost/api/v1/deployments/apiDriven' \
    -H 'Authorization: UserAccessKey 324adfe9-cc97-43fa-81c6-e248ceeb726a' \
    -H 'Content-Type: application/json' \
    -d \
    '{
        "version":{"name":"1.0",
        "application":{"externalId":"f2262e67-6516-4597-92a6-cfe537abdd00"},
        "url":"http://example.com",
        "environment":{"name":"DEV"}
    }'
```

Getting started with GraphQL in Measure

You can use GraphQL Playground in explore and test APIs interactively before using them in real scenarios. This GraphQL playground provides a convenient environment to use queries and mutations, that helps you understand the available data and how to structure requests effectively.

GraphQL is a flexible and efficient query language for APIs that allows you to request exactly the data you need. Unlike traditional REST APIs that return fixed data structures, GraphQL helps you define the shape of the response, fetch related data in a single request, and interact with data using queries (to read data) and mutations (to create or update data).

With GraphQL in Measure, you can use queries to retrieve value stream data, such as work items, deployments, commits, and other DevOps artifacts, and use mutations to create or manage resources.

Setting up GraphQL with API endpoints

Before you begin, you must create a user access key. See Managing User access keys (UAKs) on page 1820.

Follow the steps to set up and authenticate the GraphQL Integrated Development Environment (IDE) in DevOps Velocity:

1. Navigate to the Velocity instance GraphQL interface using `https://my_velocity:port/graphiql`.
2. Click the Set Headers icon.
3. Enter **Authorization** in the **Header key** field.
4. Enter the user access key in the **Value** field.

After the authentication is complete, you can start using different GraphQL APIs to validate responses and refine your queries. The GraphQL Playground also provides built-in documentation for each query and mutation, allows you to easily explore available fields, input parameters, and response structures.

After testing, you can use these APIs in your applications or leverage them to create and manage resources in Measure for bulk operations or automation scenarios without using the UI.

Mutations

Mutations are used to create new data, update existing data, and delete existing data.

Here are a few examples of mutations that you can use in Velocity.

- Create a Value Stream

  A Value Stream represents the flow of work items through a pipeline. It helps you visualize how work items move from one stage to another across various tools in the pipeline. To create a new value stream, you can use the following GraphQL mutation:

  ```
  mutation {
    createValueStream(
      name: "string"
      description: "string"
      tenantId: "_____"
      team: { id: "string", name: "string" }
      workflow: "_____"
    )
  }
  ```

- Add an environment to a pipeline

  A pipeline defines the series of steps required for deploying software, while stages represent the different phases of the pipeline (for example, build, deploy, test). To add a new stage to a pipeline, you can use the following GraphQL mutation:

  ```
  mutation {
    addPipelineStages(
      pipelineId: "Pipeline_ID",
      stages: [{ name: "<Stage_name>", type: "<Stage_type>" }]
    ) {
      _id
      name
      type
      description
      planId
      triggers {
  ```

```
      _id
      name
    }
    gates {
      _id
      name
    }
    cf_space_id
    promoteTo
    latestAutopromotionStatus
    deploymentModes
    defaultDeploymentMode
    groupsAndUsers
    solutionVersionId
  }
}
```

- Add an application to a pipeline

  To add an application to a pipeline, you can use the following mutation:

```
mutation {
  addApplications(
    pipelineId: "_____"
    applications: [{ name: "string" }]
    processes: [{ name: "string" }]
    type: "string"
  ) {
    _id
    name
    tenant_id
    description
    stages {
      _id
      name
      type
      description
      planId
      triggers
      gates
      cf_space_id
      promoteTo
      latestAutopromotionStatus
      deploymentModes
      defaultDeploymentMode
      groupsAndUsers
      solutionVersionId
      solutionVersion
    }
    applications {
      _id
      name
      type
      description
      processes
      source
      inputJobId
      stages
      subApplications
```

```
      children
      level
      external_id
      selectedChildren
      versionRegEx
    }
    synchronize
    team {
      id
      name
      tenantId
    }
    created
    stats {
      leadTime
    }
  }
}
```

Queries

You can access detailed information from Measure by using GraphQL queries.

The following are some example queries that demonstrate how to retrieve various types of data.

- Get application data by using the application ID

  You can retrieve detailed information about a specific application by using its ID and the following query:

```
{
  applicationById(id: "<Application_ID>", tenantId: "<Tenant_ID>", loadChildren: true) {
    _id
    id
    external_id
    tenant_id
    integration_id
    name
    alphaNumericName
    active
    createdAdHoc
    json_data
    type
    version
    level
    childrenIds
    children {
      _id
      id
      external_id
      tenant_id
      integration_id
      name
      alphaNumericName
      active
      createdAdHoc
      json_data
      type
```

```
      version
      level
    }
  }
}
```

- Retrieve the application environment details by tenantID

  To retrieve all unique application environment details for a given tenantID, you can use the following query::

```
{
  applicationEnvironmentsByTenantId(tenantId: "TenantID") {
    _id
    id
    external_id
    tenant_id
    name
    app_id
    external_app_id
    active
    json_data
    type
    version
    inventory {
      external_version_id
      components
    }
  }
}
```

- Retrieve application environment name by tenantID

  To retrieve all unique application environment names for a given tenantID, you can use the following query:

```
{
  uniqueApplicationEnvironmentNamesByTenantId(tenantId: "_____") {
    name
  }
}
```

How to

The topics in this section describe how to work with Measure features. Tutorials and workbooks are available for users of all experience levels.

Creating reports

Use the **Home** page to create one of the following reports.

About this task

- Custom
- Delivery Transformation
- Recent Activity

- Team Performance
- Audit

1. From the **Home** page, click **Insights > Reports**.
2. Click to **Add report**.
3. On the **New report** page, in the **Type of report** field, select the type of report from the list.
4. Enter the name of the report in the **Report name** field.
5. Enter a report description in the **Description** field.
6. Select an icon for the report in the **Icon** field.
7. Click **Add**.
   After the report is created you can immediately run, edit, and share the report with other users.

Sharing reports

You can share reports with others in your organization and who are authenticated to Measure. By default, you see only the reports that you create.

About this task

When you share a report, the users you are sharing the report with are able to access the report from their Home page. Depending on the access level that you provide, they can modify the report or only view report data.

Note: The data users can view is restricted to data from the teams of which they are a member.

You can remove access at any time by clicking the delete icon next to the email address on the **Share Report** page.

1. From the **Insights > Reports** page, select the report that you want to share.
2. Click

to copy the URL link for sharing the report.
3. Share the URL of the report to required users.

   Note: The user must be authenticated for accessing the reports.

Glossary

This glossary provides terms and definitions for the Measure software and products.

The following cross-references are used in this glossary:

- *See* refers you from a non-preferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

A

activity

Activities include releases, events, and deployment plans that you develop, start, and complete with the product.

agent

A program that runs on a remote computer and communicates with the Measure server. Agents run processes on deployment target systems.

agent relay

A communication proxy for agents that are located behind a firewall or in another network location.

API

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

Application

One or more computer programs or software components that provide a function in direct support of a specific business process or processes. See also application server.

application environment

A user-defined collection of resources that hosts an application. These application environments refer to environments that are created to be in the product.

application process

A process that is associated with an application. Unlike a component or generic process, an application process is created from application-level steps. Typically, an application process invokes component processes, and orchestrates multi-component deployments. See also component process on page 1900, generic process on page 1901, process on page 1903.

artifact

A deployable item such as a file, image, database, configuration material, or anything else that is associated with a software project. By default, artifacts are stored in CodeStation repository.

authentication realm

An authentication realm manages users and determines user identity. It retrieves information from sources such as LDAP servers and Keystone identity services.

auto task

A task that starts automatically they are eligible to run. Examples of auto task types are HCL DevOps Deploy, Continuous Delivery pipeline, email, and Slack.

B

blackout

A setting that prevents deployments and snapshots from being scheduled. Blackouts solely affect the environment and application that they are applied to.

C

change

An artifact that represents a generalized notion of a development work product, such as a task, defect report, or enhancement request. See also initiative on page 1901.

component

A representation of deployable items and the user-defined processes that operate on them, usually by deploying them.

component process

A process defined for the deployment of components.

component version

See version on page 1904.

D

dashboard

A displayed collection of user-defined data filters and metric charts. Dashboards are used to analyze project data for their quality and efficiency.

dependent task

A task depends on another task to finish before it starts. The task that a dependent task waits for is called a prerequisite task. Dependent tasks cannot start until all its prerequisite tasks are resolved. Dependent tasks are used to ensure that tasks run in the specified order.

deployment

The activities used to deliver a software project to a deployment target. Typically, you run deployments for each stage of your release lifecycle, ending with the production stage when the software becomes generally available. The activities that are performed during a deployment, called tasks, are defined in deployment plans. You start a deployment by starting one of the plan's eligible tasks. You complete a deployment by resolving all the tasks in the deployment plan.

dots

A value stream unit of work, such as a pull request, build, or deployment. Units of work can be combined, such as commits and pull requests, which significantly enhances the data quality of value streams.

duration

The time a task takes to run. Duration is measured from the time a task starts until it is resolved. When you create some task types, you can estimate its expected duration. Duration is reported in minutes.

E

environment

A collection of resources that identify the components that can be deployed by the parent application and the agents that do the work.

events

Release-related activities that are applied to releases and tracked with a calendar. You can use events to organize your releases and other time-dependent activities, such as holidays and blackouts.

execution pattern

The order in which tasks in a deployment plan become eligible to run. By default, a plan's execution pattern is sequential. Sequential tasks run in order beginning with the first task in the plan. You can combine tasks into groups and assign the parallel execution pattern to the group. Parallel tasks can run in any order and run simultaneously.

G

gate

A requirement that must be met before an application can move on to the next stage.

generic process

Processes that are designed to run outside typical component or application processing. See also application process on page 1899, component process on page 1900, process on page 1903.

group

A logical container that serves as a mechanism to grant permissions to multiple users.

I

initative

An action to take for the change occurred.

insights

A set of related features such as metric data dashboards, reports, and the application portfolio.

integration

Regular communication between Measure and external products and services. Communication with integrated products can be bidirectional.

L

lifecycle

The phases in a release. A lifecycle is a template for the stages of work in a release.

lock

A means of preventing changes being made to a release.

logical environment

A collection of physical environments on page 1902.

M

manual task

A release related task that users must manually complete. See also auto task on page 1899.

metric data

Tools, such as charts and graphs that are applied to project data to assess project quality and progress.

metric definition

A JSON object that defines expected value types and chart default settings.

milestone

A significant event in a project or process that is used to determine progress toward goals.

P

particles

See dots on page 1900.

phase

An organizational segment within a value stream containing the stages, such as "Planning," or "Development,", that define process flow within that group.

physical environment

An actual deployable location, server, hardware, and so on.

pipeline

A sequence of stages that retrieve input and run applications and jobs, such as builds, tests, and deployments.

plugin

A separately installable software module that adds function to an existing program, application, or interface.

portfolio

A collection of application information in one location where you can analyze the data with metric charts and graphs.

process

In Measure, processes define automation activities, such as deploying components.

property

A variable that specifies a value that is provided during process deployment and is associated with a specific Measure object. An individual property's value can be accessed only by certain types of related objects, such as environments, processes, and components.

Q

quality

A user-defined attribute that serves as a gate for applications to enter phases.

R

release

The delivery of multiple applications through one or more phases of a lifecycle.

resource

A user-defined construct that is based on the architectural model of Measure. A resource represents a deployment target.

role

A job function that identifies the tasks that a user can perform and the resources to which a user has access. A user can be assigned one or more roles.

S

segment

A period of time in a deployment plan. Deployment plans can group tasks into segments to specify when tasks are run relative to each other.

snapshot

A collection of specific versions of components. Typically, a snapshot represents a set of component versions that are known to work together.

stage

In the value stream, a subset of a phase that contains data from integrated tools.

swim lane

A representation of value stream data in tabular form where the columns represent value stream stages, and the rows represent users.

T

tag

A label that you can apply to tasks or releases. When applied to tasks, tags can determine task applicability for a given deployment. When applied to releases, tags can be used to organize and filter releases.

task

Represents a business-meaningful activity that has starting and ending points and a measurable duration. Durations are used to estimate deployment times. You add tasks to deployment plans. When you run a deployment, you complete the tasks in the plan

task group

A combination of two or more tasks that are associated into a group. When you form a group, you define the group's execution pattern as sequential or parallel.

team

A team is a collection of users and groups.

token

A particular message or bit pattern that signifies permission or temporary control to transmit over a network.

U

user

A representation of an account on the server. Users can be members of teams and groups. User can be created in Velocity Measure server or import users from an external authentication realm, including an LDAP, Active Directory, or SSO provider.

V

value stream

A representation of the objects. The value stream provides a strategic window into your project workflow.

version

A representation of an Deploy application snapshot.

Chapter 7. Loop Genie overview - Tech Preview

Disclaimer:

This release contains access to the Loop Genie as a Tech Preview in HCL DevOps Loop. The Tech Preview is intended for you to view the capabilities of Loop Genie offered by HCL DevOps Loop, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

Loop Genie is an AI-powered assistant integrated into DevOps Loop. It provides a unified conversational interface for retrieving information and performing actions across multiple DevOps Loop capabilities.

Loop Genie uses a ReAct (Reason and Act) execution model that dynamically determines the next action required to satisfy a request. Rather than following a fixed sequence of operations, Loop Genie evaluates results after each step and can invoke additional tools as needed until the requested task is completed.

Loop Genie supports multi-step requests that can involve multiple tools in a single interaction. For example, you can create a work item, assign it to a user, and retrieve its status within a single prompt. Loop Genie also produces formatted responses for better readability, replacing raw JSON outputs.

Loop Genie connects with built-in agents to handle prompts securely and efficiently. You can interact with Loop Genie through text or voice input, making it easier to manage teamspaces, add or remove users, create branches, edit files, and manage pull requests without navigating multiple interfaces.

Loop Genie's AI capabilities support integration with **OpenAI**, **IBM watsonx**, **Claude**, and **Gemini**. These integrations provide natural language understanding, context-aware responses, and advanced automation, allowing Loop Genie to perform tasks and retrieve information efficiently across your DevOps Loop environment.

Operational modes

Loop Genie supports the following operational modes:

Query mode

Query mode is the default mode and is designed for information retrieval and exploration. In Query mode, Loop Genie uses read-only tools and does not make changes to your DevOps Loop environment.

You can use the Query mode when you want to retrieve information, search across loops and capabilities, review work items, repositories, or metrics, and explore available resources.

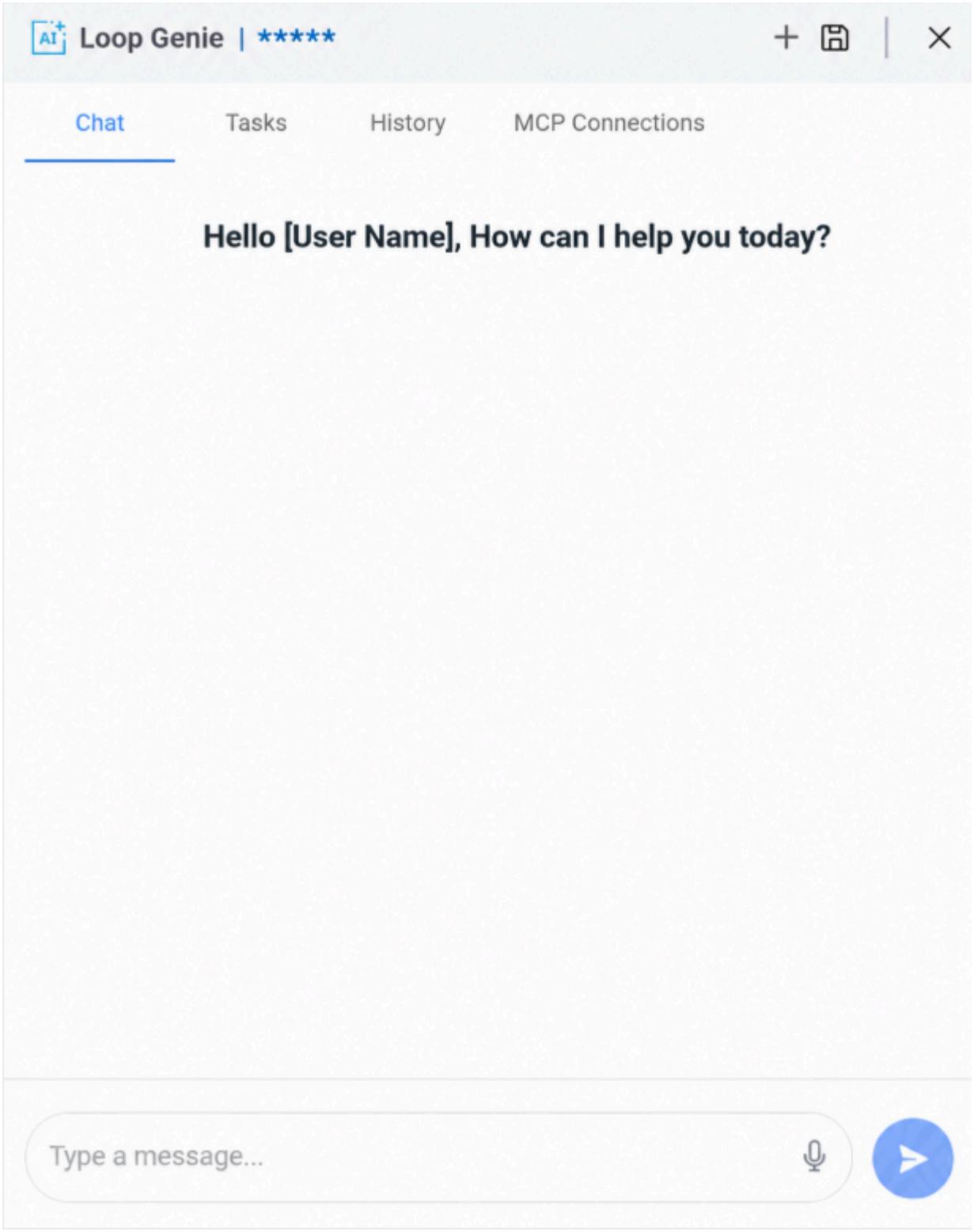
Task mode

Task mode is used for predefined or user-defined tasks that perform actions within DevOps Loop. Depending on the task configuration and assigned permissions, Loop Genie can run write operations such as creating work items, updating records, or modifying repositories.

Loop Genie workspace

Loop Genie provides a unified workspace for interacting with AI-assisted DevOps Loop capabilities. The workspace enables you to submit requests, run predefined tasks, review tool activity, and access previous conversations.

When you click Loop Genie icon in the loop card, the Loop Genie workspace is displayed with following options:



Chat

You can use the **Chat** tab to interact with Loop Genie. The Chat tab provides following:

- Text and voice input
- Streaming responses
- Tool execution visibility
- Context usage monitoring
- Conversation downloads
- New conversation creation

During execution, Loop Genie displays tool activity inline so that you can review the actions performed and the results returned by each tool.

Tasks

You can use the **Tasks** tab to access predefined and custom tasks.

Tasks provide reusable workflows that combine prompts, instructions, and tool access to automate common DevOps activities. Tasks can be categorized and marked as favorites for easier access.

History

Navigate the History tab to review previous interactions.

History is filtered by the selected loop and current user. You can revisit previous conversations, review results, rerun requests, and download conversations for future reference.

MCP Connections

You can view available MCP servers to Loop genie and enable or disable added MCP servers directly from the **MCP Connections** tab.

Monitoring Execution

The workspace displays context usage information during execution. This indicator helps you understand how much context is currently being used by the conversation.

When context usage approaches configured limits, Loop Genie can automatically compress conversation history to continue processing requests.

Stopping execution

If a request is taking longer than expected or is not producing the desired results, click Stop to interrupt the current execution.

Sample multi-step workflow execution in Loop Genie

Scenario: You want to create a new work item in DevOps Loop, assign it to a user, and then fetch its status all in a single request.

Prompt to Loop Genie:

“

Create a new story in the `Payment API` project titled ‘Implement new refund feature’, assign it to `alice@example.com`, and then get the status of the newly created work item.”

”

How Loop Genie handles the execution:

1. **Step 1 – Create Work Item**
   - Calls `plan_create_work_item` with project name, title, and optional fields.
   - Returns the new work item ID.
2. **Step 2 – Assign User**
   - Calls `plan_update_work_item` (or equivalent MCP command) to assign the work item to `abc@example.com`.
3. **Step 3 – Retrieve Status**
   - Calls `analytics_get_work_item_status` with the work item ID to fetch the current status.

Output returned by Loop Genie:

Field	Value
Work Item ID	12345
Title	Implement a new refund feature
Assigned To	abc
Status	Open

Interacting with Loop Genie

You can interact with Loop Genie, an AI assistant, to retrieve information or perform tasks across multiple capabilities and loops within DevOps Loop. The results or actions are displayed directly in the chat interface.

Before you begin

You must have completed the following tasks:

- An AI provider is configured on the **Installed Integrations** page. See AI provider integration for Loop Genie - Tech Preview on page 94.
- Created a loop. See Creating a loop on page 140.

1. Click the button.

   Result

   A chat interface is displayed with a greeting message: *"Hello (username), How can I help you today?"*
2. Perform any of the following actions:
   a. Enter your prompt or query in the chat box.
   b. Navigate to the **Tasks** tab and select the task you want Loop Genie to perform.
3. Click the **Send** icon or press **Enter**.

   Loop Genie processes your query, performs actions, or both, depending on the request. The results or actions are displayed directly in the chat interface.

   During execution, Loop Genie displays tool activity within the conversation. For requests that involve multiple tools or large result sets, information may appear progressively as processing continues. Depending on your configuration, you can review the following execution details:

   - Tool names
   - Tool parameters
   - Execution status
   - Tool results

   Note: The output format might vary between AI providers, which can affect how responses are displayed in Loop Genie.
4. **Optional:** Click the icon to download the conversation in a `.json` format for later reference.
5. **Optional:** Click to close the Loop Genie.

Results

You have interacted with Loop Genie to retrieve information or perform tasks across DevOps Loop capabilities.

Related information

Running a predefined task in Loop Genie on page 3284

Prompt references for Loop Genie on page 3286

Loop Genie tasks

Loop Genie tasks are reusable workflows that help automate common DevOps activities. A task combines instructions, prompts, and tool access into a predefined workflow that can be executed from the Tasks tab in the Loop Genie workspace.

Instead of manually entering prompts for repetitive activities, you can run a task to perform a predefined sequence of actions consistently and efficiently.

Creating a loop automatically makes a set of preloaded tasks available. The system groups these tasks into categories such as build, code, and deployment, helping you quickly find relevant workflows.

Each task is defined as a markdown file and stored in a dedicated repository associated with the loop. These task definitions are version-controlled, and you can update, extend, or customize tasks.

From the **Tasks** tab, you can perform the following actions:

- View available tasks and their descriptions
- Filter tasks by category
- Mark frequently used tasks as favorites
- Select and run tasks directly

When you select a task, it opens in the **Chat** tab where you can enter your own custom prompt and choose to run the workflow or perform specific actions within the task. Tasks provide a more structured and efficient way to interact with Loop Genie by reducing repetitive input and providing consistent and reusable workflows.

Task repository

Task definitions are stored as markdown files in the task repository associated with a loop.

When a loop is created, a task repository is automatically created to store task definitions. These definitions are version controlled and can be updated as requirements change.

Administrators and authorized users can create new tasks, modify existing tasks, and maintain task definitions by updating the files stored in the repository.

How tasks work

When you select a task, Loop Genie loads the task definition and uses the instructions and permissions associated with that task during execution. A task can include:

- Task metadata
- A description
- System instructions
- Tool permissions
- Suggested prompts
- Categories
- Starter actions

Loop Genie uses this information to determine how the task should behave and which tools can be used while processing requests.

Task execution

Tasks use Loop Genie's reasoning and action workflow to process requests. During execution, Loop Genie:

1. Evaluates the task instructions.
2. Determines the next action required.
3. Selects an appropriate tool.
4. Executes the tool.
5. Reviews the result.
6. Continues until the objective is completed.

Depending on the task configuration, multiple tools can be used during a single execution.

Running a predefined task in Loop Genie

You can run predefined tasks that are available in the **Tasks** tab in Loop Genie to run AI-assisted DevOps workflows through configured prompts and toolsets.

Before you begin

You must have completed the following tasks:

- Created a loop. See Creating a loop on page 140.
- An AI provider is configured on the **Installed Integrations** page. See AI provider integration for Loop Genie - Tech Preview on page 94.
- Verified that predefined tasks are available in the **Tasks** tab in Loop Genie.

1. Click the button.
   A chat interface is displayed.
2. Navigate to the **Tasks** tab and select a task.
   The task opens in the **Chat** tab with its details and available steps. You can choose a suggested step or enter a new instruction.
3. Select a step.

   The task runs and returns results based on the predefined System Prompt and toolsets defined in Task's markdown file.
4. Review the results.

Results

You have successfully run a predefined Loop Genie task and received results based on the configured prompts and toolsets.

Creating or modifying a task in Loop Genie

You can create a new or modify an existing Loop Genie task by defining prompts and selecting tools to automate DevOps workflows.

Before you begin

You must have created a loop.

1. Select **Control** from the **Go To** drop-down in the loop.
   The repository associated with the loop is displayed.
2. Navigate to your **TeamSpace** and then select the task repository named as `<loop_name>-Tasks`.

   When a loop is created, a repository is automatically created to store task definitions as markdown files (`.md`). For example: `<loop_name>~TASKS`. You can use this repository to add new task files or modify existing ones.

   Note: As this is a standard source control repository, you can clone it locally, make changes using your preferred editor, and push updates, or edit files directly through the UI.

3. Create a new task file or modify an existing one in the repository.
4. Define or update the task definition file with the required structure and details. Each task file contains YAML front matter that defines how the task appears and behaves.

   Table 1203. YAML front matter for Loop Genie Task

Field	Required	Description
Name	Mandatory	Specifies the name of the task.
Description	Mandatory	Provides a brief description of the task.
Icon	Optional	Specifies the icon for the task. Provide a relative path (for example, `icons/build.svg`) with respect to the current markdown file location. If not provided, a default icon is used.
Categories	Optional	Defines the category of the task. If not specified, the task is displayed under **Others**.
Tools	Optional	Lists the tools that the task can use. Tool names must match exactly. If not provided, all available tools based on the configured MCP are used.
Starters	Optional	Defines conversational prompts for common actions. Each starter must include a **label** (displayed in the UI) and a **prompt** (sent to Loop Genie). The prompt is shown as a tooltip.
System prompt	Mandatory	Defines the behavior and instructions that control how the task executes.

5. Save the file and commit your changes to the repository.

6. Open Loop Genie .
7. Navigate to the **Tasks** tab.
   The created or modified task is displayed.

Results

You have created a new or modified an existing Loop Genie task.

Prompt references for Loop Genie

You can use natural language prompts to interact with Loop Genie and perform actions across DevOps Loop capabilities.

Loop Genie uses a reasoning and action workflow to determine the tools and steps required to satisfy a request. Rather than requiring a separate command for each action, Loop Genie can process multi-step requests and coordinate actions across multiple capabilities within a single prompt.

The prompts in this topic are examples only. You can modify them to match your environment, naming conventions, and workflows.

Writing effective prompts

Describe the outcome that you want to achieve rather than each step. Loop Genie can determine the required sequence of actions and execute the appropriate tools to complete the request.

For example, instead of submitting multiple requests such as:

- Create a work item.
- Assign the work item.
- Retrieve the work item status.

Use a single request such as:

Create a story named "Implement refund feature", assign it to john@example.com, and then retrieve the work item status.

Using tool references

You can type **#** in the prompt field to view available tools from enabled MCP connections.

Examples:

- `#plan_create_work_item`
- `#control_get_pull_request`
- `#analytics_get_work_item_status`

Tool references can help direct Loop Genie to a specific capability or operation when multiple tools are available.

Prompt references

You can use the following prompts to interact with Loop Genie, which responds and performs the specified actions on your DevOps Loop platform. These prompts are for reference only and can be customized based on your environment and version.

Loop Genie supports multi-step workflows. A single prompt can involve multiple capabilities, and Loop Genie executes the required actions as part of a single request.

Table 1204. Sample prompts for DevOps Loop capabilities

Module	Scope	Prompt examples
Loop	Manage loops, projects, users, test projects, and control repositories within a loop.	• List all projects under Loop "TeamAlpha". • Add user "jane.doe@example.com" to Loop "TeamAlpha". • Attach Test Project "QA-Release1" to Loop "TeamAlpha". • Add repository "devops-repo" to Loop "TeamAlpha".
Loop Teamspaces	Manage teamspaces, teamspace users, and invitations.	• Show all teamspaces in DevOps Loop. • List all users in Teamspace "Engineering". • Invite "mark.smith@example.com" to Teamspace "Engineering".
Loop (Plan)	Manage planning activities such as components, work items, and applications.	• List all components for Plan ID "Plan123". • What work item types are supported in Plan "Plan123"? • Show work items for Project "InventoryApp". • List applications under Plan "Release2.0". • Create a Bug work item "Fix login issue" in Project "InventoryApp". • Delete work item ID "4567" from Plan "Plan123".
Loop (OpenSearch)	Search and analyze work items, story points, and statuses.	• Search for work items containing "API Bug" in Sprint "Sprint7". • Show the status of Work Item ID "789".

Table 1204. Sample prompts for DevOps Loop capabilities (continued)

Module	Scope	Prompt examples
		• How many open work items exist in Plan "Plan123"? • Show total story points for Sprint "Sprint5".
Loop (TestHub)	Manage and execute tests and test projects.	• List all TestHub projects for Loop "QA-Team". • List all tests in Project "ReleaseSmokeTests". • Run Test ID "T101" in Project "ReleaseSmokeTests". • Show results for Test ID "T101" in Project "ReleaseSmokeTests". • Get TestHub result for ID "5555".
Loop (Control)	Manage repositories, branches, tags, releases, issues, pull requests, and searches.	• Show details of the authenticated user. • List all organizations linked to my account. • Fork the repository "main-project" to my personal workspace. • Show all repositories I own. • Create a branch "feature-123" in the repository "backend-service". • List branches in repository "frontend-app". • Create release "v1.0" in repository "backend-service". • Get details of release "v1.0" in repository "frontend-app". • List all releases for the repository "api-service". • Create tag "v2.0" in repository "frontend-app". • Get details of tag "v1.0-beta" in repository "backend-service". • List tags in the repository "frontend-app". • Show commits in the repository "backend-service". • Get the content of "README.md" in the repository "frontend-app". • Update file "config.yaml" in repository "api-service". • Get issue #12 from the repository "frontend-app".

Table 1204. Sample prompts for DevOps Loop capabilities (continued)

Module	Scope	Prompt examples
		• List all issues in the repository "backend-service". • Create an issue "Fix API bug" in the repository "api-service". • Edit the comment on issue #12 in repository "frontend-app". • Show comments for issue #34. • List pull requests for the repository "backend-service". • Create a pull request from branch "feature-123" to "main" in the repository "api-service". • Search for team "DevOps" in organization "Enterprise". • Show the current Gitea MCP Server version.

Multi-step prompt examples

Loop Genie can coordinate multiple actions across capabilities within a single request as follows.

- Create a Bug work item named "Fix login issue", assign it to jane.doe@example.com, and then retrieve its status.
- Create a branch "feature-refund-api" in the repository "payment-service" and then show the branch details.
- Add user "jane.doe@example.com" to Loop "TeamAlpha" and then list all users in the loop.
- Create a pull request from the branch "feature-refund-api" to "main" and then show the pull request details.

Depending on the configuration, Loop Genie can display tool activity during execution, including tool names, parameters, execution status, and results. This information helps you understand how requests are processed and provides visibility into actions performed by Loop Genie.

Chapter 8. Troubleshooting

The troubleshooting information can help you diagnose and resolve common issues. If the issue persists after completing the applicable troubleshooting steps, gather the relevant logs and error details before contacting support. You can contact HCL Support if you are unable to troubleshoot the problem. For more information, see HCL Customer Support.

Security Considerations

This document describes the actions that you can take to ensure that your installation is secure, customize your security settings, and set up user access controls in HCL DevOps Loop.

You can find the following information about security considerations for the solution:

- Security considerations for Plan
- Security Considerations for Test
- Security Considerations for Deploy
- Security settings for Measure and Release and Considerations for GDPR readiness

Notices

This document provides information about copyright, trademarks, terms and conditions for the product documentation.

© Copyright IBM Corporation 2000, 2016 / © Copyright HCL Technologies Limited 2016, 2024

This information was developed for products and services offered in the US.

HCL® may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL® representative for information on the products and services currently available in your area. Any reference to an HCL® product, program, or service is not intended to state or imply that only that HCL® product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL® intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL® product, program, or service.

HCL® may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL® Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL® may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL® websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL® product and use of those websites is at your own risk.

HCL® may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL® under terms of the HCL® Customer Agreement, HCL® International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL® products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL® has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL® products. Questions on the capabilities of non-HCL® products should be addressed to the suppliers of those products.

Statements regarding the future direction or intent of HCL® are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to HCL®, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. HCL®, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. HCL® shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from HCL Ltd. Sample Programs.
© Copyright HCL Ltd. 2000, 2022.

Trademarks

HCL®, the HCL® logo, and hcl.com® are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of HCL® or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the HCL® website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of HCL®.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of HCL®.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL® reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL®, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL® MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

A

access key 1811, 1820, 3029, 3038
accessibility
  DevOps Loop
    disability 36, 708
    keyboard 36, 708
  Test
    disability 36, 708
    keyboard 36, 708
acquire-lock 2978
add metrics 3081
add user action 806
Add user action for this element 796
Add user action in the test editor 806
add-inventory-status 2964
addAgentPoolToTeam
  udclient command 2596
addAgentServerPin
  udclient command 2596
addAgentToPool
  udclient command 2597
addAgentToTeam
  udclient command 2598
addApplicationToTeam
  udclient command 2598
addComponentResourcePropDef
  udclient command 2599
addComponentTemplateToTeam
  udclient command 2600
addComponentToApplication
  udclient command 2600
addComponentToTeam
  udclient command 2601
addEnvironmentBaseResource
  udclient command 2601
addEnvironmentProperty
  udclient command 2602
addEnvironmentToTeam
  udclient command 2603
addGroupToTeam
  udclient command 2604
addJavaPackage
  udclient command 2604
addPostProcessingScriptToTeam
  udclient command 2605
addProcessToTeam
  udclient command 2605
addResourceToTeam
  udclient command 2606
addStatusToSnapshot
  udclient command 2607
addTagToAgent
  udclient command 2608
addTagToApplication
  udclient command 2608
addTagToComponent
  udclient command 2609
addTagToResource
  udclient command 2610
addUserToGroup
  udclient command 2611
addUserToTeam
  udclient command 2612
addVersionFiles
  udclient command 2612
addVersionLink
  udclient command 2613
addVersionLinkWithName
  udclient command 2614
addVersionStatus
  udclient command 2615
addVersionToSnapshot
  udclient command 2616
addWebhookToApplication
  udclient command 2616
addWebhookToApplicationWithCredentials
  udclient command 2617
addWebhookToProcess
  udclient command 2618
addWebhookToProcessWithCredentials
  udclient command 2619
addZInventoryEntries
  udclient command 2620
admin password 1819, 3037
administering 299, 2057
administration 1799, 1799, 3017, 3017
advanced tutorials 1984
agent 2095
agent limit 2098
agent pools 342, 344, 345, 346
  creating 2106
  managing 2106
agent workspace 349
agentcomm.server.uri 2097
agentcomm.uri 2097
agents 300
alerts 3103
Ansible tasks 1869, 3241
API 1811, 1874, 3029, 3141, 3246
API authentication 1820, 3038
API-driven pipeline 3141
application deployments
  REST commands 1998
application processes
  running in target environments 2525
application versions 3145
applications
  creating 2345
apply-configuration 2934
approval 3145, 3153
approval process 2388, 2388
approvals 1831, 1832, 2388, 2388, 3203, 3203
approver rules 3145
approvers 2388
AppScan Enterprise report 3185
archive 1834, 1834, 1840, 3205, 3206, 3211
archiveVersion
  udclient command 2620
archiving 1882, 3254
Argo Workflow Engine 1874, 3246
artifacts 358
ASE report 3185
authentication 1813, 1815, 1817, 1817, 3031, 3033, 3035, 3035
authorization 1811, 1815, 1817, 3029, 3033, 3035
auto-generate 1820, 3038
autoDiscoverResource
  udclient command 2621
Automation
  overview 30
average wait time 3103

B

basic tutorials 1914, 1998
blue-green deployments 2441
blue-green deplyment 1984
blue-green topology 1984
blue-green tutorial 1984
bottleneck algorithm 3098
bottleneck detection 3098
build life 361
Build System Toolkit
 2353, 2353
builds/apidriven 3141
bulkAddAgentServerPin
  udclient command 2621
bulkGetAgentJavaHomes
  udclient command 2622
bulkGetPreferredJavaPackageForAgent
  udclient command 2623
bulkInstallAgentJava
  udclient command 2623
bulkRemoveAgentServerPin
  udclient command 2624
bulkSetAgentJavaHome
  udclient command 2624
bulkSetAgentKeypairPolicy
  udclient command 2625
bulkUninstallAgentJava
  udclient command 2626
burndwon chart 3194

C

calendar 1800, 3018
canary deployment 1984
canary deployments 2441
cancelApplicationPipelineExecution
  udclient command 2626
cancelApplicationProcessRequest
  udclient command 2627
cancelComponentProcessRequest
  udclient command 2627
cancelMaintenanceBlackoutSeries
  udclient command 2627
cancelMaintenanceBlackoutSeriesEvent
  udclient command 2628
cancelProcessRequest
  udclient command 2628
cancelScheduledDeploymentSeries
  udclient command 2629
cancelScheduledDeploymentSeriesEvent
  udclient command 2629
cancelServerKeystoreImportTask
  udclient command 2630
cancelWorkflow
  udclient command 2630
card view layout 3081
Change
  Classification 976
  Dataset 976
  Encrypt 976
Change Classification
  Dataset 973
change requests 1831, 1832, 3203, 3203
charts 3170
cleanup 327, 328
closeTask
  udclient command 2631
code editor 3067
command-line commands
  addAgentPoolToTeam 2596
  addAgentServerPin 2596

addAgentToPool 2597
addAgentToTeam 2598
addApplicationToTeam 2598
addComponentResourcePropDef 2599
addComponentTemplateToTeam 2600
addComponentToApplication 2600
addComponentToTeam 2601
addEnvironmentBaseResource 2601
addEnvironmentProperty 2602
addEnvironmentToTeam 2603
addGroupToTeam 2604
addJavaPackage 2604
addPostProcessingScriptToTeam 2605
addProcessToTeam 2605
addResourceToTeam 2606
addStatusToSnapshot 2607
addTagToAgent 2608
addTagToApplication 2608
addTagToComponent 2609
addTagToResource 2610
addUserToGroup 2611
addUserToTeam 2612
addVersionFiles 2612
addVersionLink 2613
addVersionLinkWithName 2614
addVersionStatus 2615
addVersionToSnapshot 2616
addWebhookToApplication 2616
addWebhookToApplicationWithCredentials 2617
addWebhookToProcess 2618
addWebhookToProcessWithCredentials 2619
addZInventoryEntries 2620
archiveVersion 2620
autoDiscoverResource 2621
bulkAddAgentServerPin 2621
bulkGetAgentJavaHomes 2622
bulkGetPreferredJavaPackageForAgent 2623
bulkInstallAgentJava 2623
bulkRemoveAgentServerPin 2624
bulkSetAgentJavaHome 2624
bulkSetAgentKeypairPolicy 2625
bulkUninstallAgentJava 2626
cancelApplicationPipelineExecution 2626
cancelApplicationProcessRequest 2627
cancelComponentProcessRequest 2627
cancelMaintenanceBlackoutSeries 2627
cancelMaintenanceBlackoutSeriesEvent 2628
cancelProcessRequest 2628
cancelScheduledDeploymentSeries 2629
cancelScheduledDeploymentSeriesEvent 2629
cancelServerKeystoreImportTask 2630
cancelWorkflow 2630
closeTask 2631
completeServerKeystoreImportTask 2631
copyResource 2632
copyVersion 2633
createAgentPool 2635
createApplication 2636
createApplicationFromTemplate 2637
createApplicationPipeline 2638
createApplicationProcess 2639
createApplicationTemplate 2643
createAuthenticationRealm 2645
createAuthorizationRealm 2647
createAuthToken 2644
createBlueprintDesignerIntegration 2649
createComponent 2649
createComponentProcess 2652
createComponentTemplate 2657
createDeploymentTrigger 2657
createEnvironment 2658
createEnvironmentFromTemplate 2660
createEnvironmentTemplate 2661
createGenericProcess 2662
createGroup 2665
createMaintenanceBlackout 2666
createManyVersions 2666
createPostProcessingScript 2668
createResource 2669
createResourceTemplate 2671
createResourceTemplateWithAgentPrototypes 2672
createSnapshot 2673
createSnapshotOfEnvironment 2674
createStatus 2675
createTeam 2676
createUser 2677
createVaultAppRole 2678
createVaultCertificate 2679
createVaultLdapUser 2679
createVaultSecretStore 2680
createVersion 2680
createWebhookTemplate 2681
deleteAgent 2682
deleteAgentPool 2683
deleteAgentProperty 2683
deleteAllDesiredInventoryVersions 2684
deleteApplication 2685
deleteApplicationPipeline 2685
deleteApplicationProcess 2685
deleteApplicationTemplate 2686
deleteAuthenticationRealm 2686
deleteAuthorizationRealm 2687
deleteBlueprintDesignerIntegration 2687
deleteComponent 2688
deleteComponentVersionPropDef 2688
deleteDeploymentTrigger 2689
deleteEnvironment 2689
deleteEnvironmentGate 2690
deleteEnvironmentTemplate 2690
deleteEnvironmentTemplateGate 2691
deleteFrontendServer 2691
deleteGroup 2692
deleteJavaPackage 2692
deleteMaintenanceBlackout 2693
deletePostProcessingScript 2693
deleteRedundantDesiredInventoryVersions 2694
deleteResource 2694
deleteResourceProperty 2695
deleteResourceRoleProperty 2696
deleteSecretStore 2697
deleteServerKeystoreEntry 2697
deleteSnapshot 2697
deleteStatus 2698
deleteTeam 2699
deleteUser 2699
deleteVaultAppRole 2700
deleteVaultCertificate 2700
deleteVaultLdapUser 2701
deleteVersion 2701
deleteWebhookTemplate 2702
downloadJavaPackage 2702
downloadVersionArtifacts 2703
executeApplicationPipeline 2704
exportApplication 2704
exportApplicationProcess 2705
exportComponent 2705
exportComponentProcess 2706
exportComponentTemplate 2706
exportComponentTemplateProcess 2707
exportDetailedGroup 2708
exportGenericProcess 2708
exportGroup 2709
exportSnapshot 2710
finishedImporting 2710
forceCancelWorkflow 2711
forceRenameAgent 2711
forgetAgentCert 2712
getAgent 2712
getAgentJavaHomes 2714
getAgentMaintenanceTask 2714
getAgentMaintenanceTasks 2714
getAgentPool 2715
getAgentPools 2716
getAgentProperty 2717
getAgentPrototypes 2718
getAgents 2718
getAgentStatuses 2718
getAllApplicationProcessesForApplication 2719
getAllComponentProcessRequestCount 2720
getAllComponentsForTemplate 2720
getAllLatestDesiredInventory 2721
getAllProcesses 2721
getAllTeams 2722
getAllUsersInRoleForTeam 2725
getAllUsersWithEmail 2725
getApplication 2726
getApplicationPipeline 2727
getApplicationPipelineExecution 2728
getApplicationPipelineExecutions 2728
getApplicationPipelines 2729
getApplicationProcess 2729
getApplicationProcessExecution 2730
getApplicationProcessRequest 2733
getApplicationProcessRequestProperties 2733
getApplicationProcessRequestStatus 2734
getApplicationProcessRequestVersions 2736
getApplicationProcessUnfilledProperties 2736
getApplicationProperties 2737
getApplicationProperty 2738
getApplications 2740
getApplicationsProcessRequests 2741
getApplicationsProcessRequestsCount 2744
getApplicationTemplate 2739
getApplicationTemplatePropDefs 2739
getApplicationTemplates 2740
getApprovalTasks 2745
getAuthenticationRealm 2748
getAuthenticationRealms 2748
getAuthorizationRealm 2749
getAuthorizationRealms 2749
getAuthToken 2746
getAuthTokens 2746
getAuthTokensForUser 2747
getAvailablePlugins 2749
getBlueprintNodePropertiesTemplate 2750
getBlueprintsInApplication 2750
getComponent 2751
getComponentEnvironmentProperties 2754
getComponentProcess 2755
getComponentProcessRequest 2756
getComponentProcessRequestCount 2757
getComponentProcessRequests 2757
getComponentProperties 2758
getComponentProperty 2759
getComponents 2762
getComponentsInApplication 2764
getComponentTemplatePropDefs 2759
getComponentTemplates 2760
getComponentVersionPropDefs 2760
getComponentVersions 2761
getConfigurableAgentLicenseTypes 2766
getCurrentActivityList 2766
getDatabaseLatency 2766

getDeployment Trigger 2766
getDeployment Triggers 2767
getEnvironment 2767
getEnvironmentBaseResources 2769
getEnvironmentCompliancy 2770
getEnvironmentGate 2770
getEnvironmentProperties 2771
getEnvironmentProperty 2772
getEnvironmentResourceTree 2773
getEnvironmentsInApplication 2775
getEnvironmentTemplate 2773
getEnvironmentTemplateGate 2774
getEnvironmentTemplatePropDefs 2774
getExternalApprovalProcess 2776
getExternalApprovalProcesses 2776
getFirstUserWithEmail 2776
getFrontendServers 2777
getHighWatermarks 2777
getInstalledPlugins 2778
getInventoryChangesForSnapshot 2778
getJavaPackage 2778
getJavaPackages 2779
getJavaPackagesForAgent 2779
getLatestDesiredInventory 2780
getLatestSnapshotForEnvironment 2781
getLLMVendorModels 2780
getLLMVendors 2780
getLoggingConfig 2782
getMaintenanceBlackouts 2782
getOpenIDConnectRealms 2783
getOverlappingArtifacts 2783
getPostProcessingScript 2784
getPostProcessingScripts 2784
getPreferredJavaPackageForAgent 2785
getProcessRequest 2785
getProcessRequests 2785
getRelays 2786
getRelayStatuses 2786
getResource 2787
getResourceInventoryEntriesForComponent 2788
getResourceProperties 2789
getResourceProperty 2790
getResourceRoleProperties 2790
getResourceRoleProperty 2791
getResources 2793
getResourcesInRecycleBin 2795
getResourcesWithComponentVersion 2795
getResourceTemplate 2791
getResourceTemplateChildren 2793
getSecretStore 2796
getSecretStores 2797
getServerKeystoreImportTask 2797
getServerKeystoreInfo 2798
getSnapshot 2798
getSnapshotConfiguration 2800
getSnapshotsInApplication 2801
getSnapshotVersions 2800
getStatus 2802
getStatuses 2804
getStatusList 2803
getSystemConfiguration 2805
getSystemProperties 2807
getSystemProperty 2807
getTagsForType 2808
getTagsOnAgent 2808
getTagsOnApplication 2809
getTagsOnComponent 2810
getTagsOnResource 2811
getTasksForUser 2812
getTeam 2813
getUser 2814
getUserByEmail 2815
getUsersThatCanRunDeploymentTriggers 2816
getVaultAppRole 2817
getVaultAppRoles 2817
getVaultCertificate 2817
getVaultCertificates 2818
getVaultLdapUser 2818
getVaultLdapUsers 2819
getVersionId 2819
getVersionLink 2820
getVersionLinks 2821
getVersionProperties 2822
getWebhookTemplate 2823
getWebhookTemplates 2823
getWorkflowTraceLogs 2824
getZOSVersionCumulativeManifest 2824
getZOSVersionDeployManifest 2825
getZOSVersionsWithCommitId 2825
importApplication 2826
importApplicationProcess 2827
importComponent 2828
importComponentProcess 2829
importComponentTemplate 2830
importComponentTemplateProcess 2831
importGenericProcess 2831
importGroup 2832
importLdapUser 2833
importSnapshot 2834
importVersions 2835
installAgent 2836
installAgentJava 2837
installLocalPlugin 2837
installPlugin 2838
listArtifacts 2838
loadEnvironmentCompliancy 2840
lockSnapshotConfiguration 2841
lockSnapshotVersions 2841
lockUser 2842
login 2842
logout 2843
moveResource 2843
pauseWorkflow 2845
prepareServerKeystoreImportTask 2845
previewUserLockEvent 2846
provisionEnvironment 2846
removeAgentFromPool 2849
removeAgentFromTeam 2850
removeAgentPoolFromTeam 2850
removeAgentServerPin 2851
removeApplicationFromTeam 2852
removeComponentFromApplication 2852
removeComponentFromTeam 2852
removeComponentResourcePropDef 2853
removeComponentTemplateFromTeam 2853
removeEnvironmentBaseResource 2854
removeEnvironmentFromTeam 2855
removeEnvironmentProperty 2855
removeGroupFromRoleOnTeam 2856
removeInventoryForUnmappedComponents 2856
removeProcessFromTeam 2857
removeResourceFromTeam 2857
removeTagFromAgent 2858
removeTagFromApplication 2859
removeTagFromComponent 2859
removeTagFromResource 2860
removeUserFromGroup 2860
removeUserFromRoleOnTeam 2861
removeVersionFromSnapshot 2862
removeVersionLink 2862
removeVersionLinkByName 2863
removeVersionStatus 2863
removeWebhookFromApplication 2864
removeWebhookFromProcess 2865
repeatApplicationProcessRequest 2865
requestApplicationProcess 2866
requestProcess 2868
restartAgent 2870
restoreArtifacts 2870
restoreFromRecycleBin 2871
resumeWorkflow 2871
revokeAuthToken 2872
rollbackZInventoryEntries 2873
saveEnvironmentGate 2873
saveEnvironmentTemplateGate 2874
savePostProcessingScript 2875
searchZInventoryEntries 2876
setAgentJavaHome 2876
setAgentKeypairPolicy 2877
setAgentLicenseType 2877
setAgentProperty 2878
setAgentUnavailable 2879
setApplicationProperty 2879
setComponentEnvironmentProperty 2880
setComponentProperty 2881
setComponentVersionPropDef 2882
setEnvironmentProperty 2883
setLoggingConfig 2884
setPreferredJavaPackage 2885
setResourceProperty 2886
setResourceRoleProperty 2887
setSystemConfiguration 2887
setSystemProperty 2891
setVersionProperty 2892
shutdownAgent 2893
testAgent 2893
testVaultAppRole 2894
testVaultCertificate 2894
testVaultLdapUser 2895
uninstallAgentJava 2896
unlockUser 2896
unsetPreferredJavaPackage 2897
updateAgent 2897
updateApplicationPipeline 2898
updateApplicationTemplate 2898
updateApplicationTemplatePropDefs 2900
updateAuthenticationRealm 2901
updateAuthorizationRealm 2902
updateComponent 2904
updateComponentResourcePropDef 2905
updateComponentTemplatePropDefs 2906
updateEnvironment 2907
updateEnvironmentProperty 2908
updateEnvironmentTemplate 2909
updateEnvironmentTemplatePropDefs 2911
updateExternalApprovalProcess 2912
updateMaintenanceBlackout 2912
updateResource 2913
updateResourceTemplate 2915
updateRoleActions 2916
updateTeam 2916
updateUser 2917
updateUsersInAuthRealm 2918
updateVaultAppRole 2918
updateVaultCertificate 2919
updateVaultLdapUser 2919
updateVaultSecretStore 2920
updateWebhookForApplication 2921
updateWebhookForApplicationWithCredentials 2922
updateWebhookForProcess 2923
updateWebhookForProcessWithCredentials 2923
updateWebhookTemplate 2924
upgradeAgent 2925

completeServerKeystoreImportTask
udclient command 2631
components
creating overview 2252
Configure
Dataset 973
configured 1881, 3253
containers 1874, 3246
copy deployment plan 1839, 3210
copyResource
udclient command 2632
copyVersion
udclient command 2633
create 1833, 1897, 3205, 3269
Create
Classification 976
create dashboard 3164
create plugins 1884, 3256
create version 3141
createAgentPool
udclient command 2635
createApplication
udclient command 2636
createApplicationFromTemplate
udclient command 2637
createApplicationPipeline
udclient command 2638
createApplicationProcess
udclient command 2639
createApplicationTemplate
udclient command 2643
createAuthenticationRealm
udclient command 2645
createAuthorizationRealm
udclient command 2647
createAuthToken
udclient command 2644
createBlueprintDesignerIntegration
udclient command 2649
createComponent
udclient command 2649
createComponentProcess
udclient command 2652
createComponentTemplate
udclient command 2657
createDeploymentTrigger
udclient command 2657
createEnvironment
udclient command 2658
createEnvironmentFromTemplate
udclient command 2660
createEnvironmentTemplate
udclient command 2661
createGenericProcess
udclient command 2662
createGroup
udclient command 2665
createMaintenanceBlackout
udclient command 2666
createManyVersions
udclient command 2666
createPostProcessingScript
udclient command 2668
createResource
udclient command 2669
createResourceTemplate
udclient command 2671
createResourceTemplateWithAgentPrototypes
udclient command 2672
createSnapshot
udclient command 2673
createSnapshotOfEnvironment
udclient command 2674
createStatus
udclient command 2675
createTeam
udclient command 2676
createUser
udclient command 2677
createVaultAppRole
udclient command 2678
createVaultCertificate
udclient command 2679
createVaultLdapUser
udclient command 2679
createVaultSecretStore
udclient command 2680
createVersion
udclient command 2680
createWebhookTemplate
udclient command 2681
creating deployment plans 1837, 3208
Creating reports 1897, 3269
creating tasks 1847, 3219
creating value streams 3051, 3055
curl 3091
custom metrics 3083, 3091
customizing value streams 3078
customizing workflows 3059
cycle time 3059, 3170
cycle time table 3175
cycle-time 3059
cycle-time table 3175

D

dashboard 3164, 3165, 3170
dashboards 3164
data explorer 3163, 3164, 3164
Dataset 973
datasets
encrypting 969
removing encryption 971
debug 1822, 3040
delay tasks 1856, 3228
Delete
Classification 976
deleteAgent
udclient command 2682
deleteAgentPool
udclient command 2683
deleteAgentProperty
udclient command 2683
deleteAllDesiredInventoryVersions
udclient command 2684
deleteApplication
udclient command 2685
deleteApplicationPipeline
udclient command 2685
deleteApplicationProcess
udclient command 2685
deleteApplicationTemplate
udclient command 2686
deleteAuthenticationRealm
udclient command 2686
deleteAuthorizationRealm
udclient command 2687
deleteBlueprintDesignerIntegration
udclient command 2687
deleteComponent
udclient command 2688
deleteComponentVersionPropDef
udclient command 2688
deleteDeploymentTrigger
udclient command 2689
deleteEnvironment
udclient command 2689
deleteEnvironmentGate
udclient command 2690
deleteEnvironmentTemplate
udclient command 2690
deleteEnvironmentTemplateGate
udclient command 2691
deleteFrontendServer
udclient command 2691
deleteGroup
udclient command 2692
deleteJavaPackage
udclient command 2692
deleteMaintenanceBlackout
udclient command 2693
deletePostProcessingScript
udclient command 2693
deleteRedundantDesiredInventoryVersions
udclient command 2694
deleteResource
udclient command 2694
deleteResourceProperty
udclient command 2695
deleteResourceRoleProperty
udclient command 2696
deleteSecretStore
udclient command 2697
deleteServerKeystoreEntry
udclient command 2697
deleteSnapshot
udclient command 2697
deleteStatus
udclient command 2698
deleteTeam
udclient command 2699
deleteUser
udclient command 2699
deleteVaultAppRole
udclient command 2700
deleteVaultCertificate
udclient command 2700
deleteVaultLdapUser
udclient command 2701
deleteVersion
udclient command 2701
deleteWebhookTemplate
udclient command 2702
deleting integrations 1882, 3254
dependencies 1848, 3220
deploy 1840, 3212
deploying applications
REST commands 1998
deployment plan 1836, 1840, 3207, 3211
deployment plan create 1837, 3208
deployment plan properties 1835, 1871, 3207, 3243
deployment plan templates 1839, 3211
deployments
running 2525
deployments/apidriven 3141
Dev cycle time table 3175
DevOps Deploy
getting started 1904
DevOps Deploy integrations 1890, 3262
DevOps Loop
accessibility 36
default security administration 81
disability 36

keyboard 36
DevOps query language 3049, 3084, 3117
DevOps query language metrics 3083, 3091
DevOps Research and Assessment 3094
disable integration 1881, 3253
Docker 1884, 3256
Docker Compose 1874, 3246
DockerHub 1874, 3246
DORA metrics 3094
dots 3049, 3059, 3080, 3103, 3103, 3113
downloadJavaPackage
udclient command 2702
downloadVersionArtifacts
udclient command 2703
DQL 3083, 3091, 3117
dynamic swim lanes 3113

E

Edit
Classification 976
education 2054
Elasticsearch 2556
ELK stack 2556
email approval 2388
email server 1800, 3017
email tasks 1856, 3228
environments
associating with resources 2345
events 1834, 3206
executeApplicationPipeline
udclient command 2704
exportApplication
udclient command 2704
exportApplicationProcess
udclient command 2705
exportComponent
udclient command 2705
exportComponentProcess
udclient command 2706
exportComponentTemplate
udclient command 2706
exportComponentTemplateProcess
udclient command 2707
exportDetailedGroup
udclient command 2708
exportGenericProcess
udclient command 2708
exportGroup
udclient command 2709
exporting applications
REST commands 2008
exportSnapshot
udclient command 2710
extending product function 1874, 2556, 3246
extending product functions 401
External YAML configuration file 3125, 3125

F

favorites 3164
finishedImporting
udclient command 2710
for-each-agent 2936
for-each-resource-tag 2938
forceCancelWorkflow
udclient command 2711
forceRenameAgent
udclient command 2711
forgetAgentCert
udclient command 2712
fromIntegrationName 3080

G

gate approval 3153
gate approve 3145
gate respond 3145
gate rule 3153
gate rules 3145
gates 3059, 3145
generic-manual-task 2979
getAgent
udclient command 2712
getAgentJavaHomes
udclient command 2714
getAgentMaintenanceTask
udclient command 2714
getAgentMaintenanceTasks
udclient command 2714
getAgentPool
udclient command 2715
getAgentPools
udclient command 2716
getAgentProperty
udclient command 2717
getAgentPrototypes
udclient command 2718
getAgents
udclient command 2718
getAgentStatuses
udclient command 2718
getAllApplicationProcessesForApplication
udclient command 2719
getAllComponentProcessRequestCount
udclient command 2720
getAllComponentsForTemplate
udclient command 2720
getAllLatestDesiredInventory
udclient command 2721
getAllProcesses
udclient command 2721
getAllTeams
udclient command 2722
getAllUsersInRoleForTeam
udclient command 2725
getAllUsersWithEmail
udclient command 2725
getApplication
udclient command 2726
getApplicationPipeline
udclient command 2727
getApplicationPipelineExecution
udclient command 2728
getApplicationPipelineExecutions
udclient command 2728
getApplicationPipelines
udclient command 2729
getApplicationProcess
udclient command 2729
getApplicationProcessExecution
udclient command 2730
getApplicationProcessRequest
udclient command 2733
getApplicationProcessRequestProperties
udclient command 2733
getApplicationProcessRequestStatus
udclient command 2734
getApplicationProcessRequestVersions
udclient command 2736
getApplicationProcessUnfilledProperties
udclient command 2736
getApplicationProperties
udclient command 2737
getApplicationProperty
udclient command 2738
getApplications
udclient command 2740
getApplicationsProcessRequests
udclient command 2741
getApplicationsProcessRequestsCount
udclient command 2744
getApplicationTemplate
udclient command 2739
getApplicationTemplatePropDefs
udclient command 2739
getApplicationTemplates
udclient command 2740
getApprovalTasks
udclient command 2745
getAuthenticationRealm
udclient command 2748
getAuthenticationRealms
udclient command 2748
getAuthorizationRealm
udclient command 2749
getAuthorizationRealms
udclient command 2749
getAuthToken
udclient command 2746
getAuthTokens
udclient command 2746
getAuthTokensForUser
udclient command 2747
getAvailablePlugins
udclient command 2749
getBlueprintNodePropertiesTemplate
udclient command 2750
getBlueprintsInApplication
udclient command 2750
getComponent
udclient command 2751
getComponentEnvironmentProperties
udclient command 2754
getComponentProcess
udclient command 2755
getComponentProcessRequest
udclient command 2756
getComponentProcessRequestCount
udclient command 2757
getComponentProcessRequests
udclient command 2757
getComponentProperties
udclient command 2758
getComponentProperty
udclient command 2759
getComponents
udclient command 2762
getComponentsInApplication
udclient command 2764
getComponentTemplatePropDefs
udclient command 2759
getComponentTemplates
udclient command 2760
getComponentVersionPropDefs
udclient command 2760
getComponentVersions
udclient command 2761
getConfigurableAgentLicenseTypes
udclient command 2766
getCurrentActivityList
udclient command 2766
getDatabaseLatency
udclient command 2766
getDeploymentTrigger
udclient command 2766
getDeploymentTriggers

udclient command 2767
getEnvironment
udclient command 2767
getEnvironmentBaseResources
udclient command 2769
getEnvironmentCompliancy
udclient command 2770
getEnvironmentGate
udclient command 2770
getEnvironmentProperties
udclient command 2771
getEnvironmentProperty
udclient command 2772
getEnvironmentResourceTree
udclient command 2773
getEnvironmentsInApplication
udclient command 2775
getEnvironmentTemplate
udclient command 2773
getEnvironmentTemplateGate
udclient command 2774
getEnvironmentTemplatePropDefs
udclient command 2774
getExternalApprovalProcess
udclient command 2776
getExternalApprovalProcesses
udclient command 2776
getFirstUserWithEmail
udclient command 2776
getFrontendServers
udclient command 2777
getHighWatermarks
udclient command 2777
getInstalledPlugins
udclient command 2778
getInventoryChangesForSnapshot
udclient command 2778
getJavaPackage
udclient command 2778
getJavaPackages
udclient command 2779
getJavaPackagesForAgent
udclient command 2779
getLatestDesiredInventory
udclient command 2780
getLatestSnapshotForEnvironment
udclient command 2781
getLLMVendorModels
udclient command 2780
getLLMVendors
udclient command 2780
getLoggingConfig
udclient command 2782
getMaintenanceBlackouts
udclient command 2782
getOpenIDConnectRealms
udclient command 2783
getOverlappingArtifacts
udclient command 2783
getPostProcessingScript
udclient command 2784
getPostProcessingScripts
udclient command 2784
getPreferredJavaPackageForAgent
udclient command 2785
getProcessRequest
udclient command 2785
getProcessRequests
udclient command 2785
getRelays
udclient command 2786
getRelayStatuses
udclient command 2786
getResource
udclient command 2787
getResourceInventoryEntriesForComponent
udclient command 2788
getResourceProperties
udclient command 2789
getResourceProperty
udclient command 2790
getResourceRoleProperties
udclient command 2790
getResourceRoleProperty
udclient command 2791
getResources
udclient command 2793
getResourcesInRecycleBin
udclient command 2795
getResourcesWithComponentVersion
udclient command 2795
getResourceTemplate
udclient command 2791
getResourceTemplateChildren
udclient command 2793
getSecretStore
udclient command 2796
getSecretStores
udclient command 2797
getServerKeystoreImportTask
udclient command 2797
getServerKeystoreInfo
udclient command 2798
getSnapshot
udclient command 2798
getSnapshotConfiguration
udclient command 2800
getSnapshotsInApplication
udclient command 2801
getSnapshotVersions
udclient command 2800
getStatus
udclient command 2802
getStatuses
udclient command 2804
getStatusList
udclient command 2803
getSystemConfiguration
udclient command 2805
getSystemProperties
udclient command 2807
getSystemProperty
udclient command 2807
getTagsForType
udclient command 2808
getTagsOnAgent
udclient command 2808
getTagsOnApplication
udclient command 2809
getTagsOnComponent
udclient command 2810
getTagsOnResource
udclient command 2811
getTasksForUser
udclient command 2812
getTeam
udclient command 2813
getting started
guide 30
introduction 1904
getUser
udclient command 2814
getUserByEmail
udclient command 2815
getUsersThatCanRunDeploymentTriggers
udclient command 2816
getVaultAppRole
udclient command 2817
getVaultAppRoles
udclient command 2817
getVaultCertificate
udclient command 2817
getVaultCertificates
udclient command 2818
getVaultLdapUser
udclient command 2818
getVaultLdapUsers
udclient command 2819
getVersionId
udclient command 2819
getVersionLink
udclient command 2820
getVersionLinks
udclient command 2821
getVersionProperties
udclient command 2822
getWebhookTemplate
udclient command 2823
getWebhookTemplates
udclient command 2823
getWorkflowTraceLogs
udclient command 2824
getZOSVersionCumulativeManifest
udclient command 2824
getZOSVersionDeployManifest
udclient command 2825
getZOSVersionsWithCommitId
udclient command 2825
git credential manager 233
global DQL query 3103
glossary 1898, 3270
Grant
access 976
Classification 976
graphs 3170
groups 323, 1819, 3037
roles 1801, 3019
guide
getting started 30

H

header tasks 1857, 3229
header-based SSO 1817, 3035
how to 1897, 3269

I

IBM Cloud Private 1831, 1871, 3203, 3243
import tasks 1837, 1842, 3209, 3213
importApplication
udclient command 2826
importApplicationProcess
udclient command 2827
importComponent
udclient command 2828
importComponentProcess
udclient command 2829
importComponentTemplate
udclient command 2830
importComponentTemplateProcess
udclient command 2831
importGenericProcess
udclient command 2831
importGroup
udclient command 2832

importLdapUser
udclient command 2833
importSnapshot
udclient command 2834
importVersions
udclient command 2835
incompatible versions 1875, 3247
initial sync date 1882, 3254
Insights 3163, 3164, 3164, 3165, 3184
insights reports 3184
install-component 2940, 2942
installAgent
udclient command 2836
installAgentJava
udclient command 2837
installations 1875, 1876, 1882, 1882, 3247, 3248, 3254, 3254
installLocalPlugin
udclient command 2837
installPlugin
udclient command 2838
integration 1874, 1884, 3078, 3246, 3256
integration logs 1881, 3253
integrations 1875, 1876, 1881, 1882, 1882, 3247, 3248, 3253, 3254, 3254
intermediate tutorials 1933
items 3103

J

JavaScript 3067
JavaScript Object Notation 240
Jenkins (Legacy) 1888, 3260
Jenkins integrations 1886, 1888, 3258, 3260
Jenkins tasks 1858, 3230
JMS agent 2096
JMS communication 238
job 375
job step
add
378
job step
arrange
378
job step
copy
378
job step
delete
383
job step
disable
383
jobs
code-as-file
380
jobs
step
378, 383
join 2945
join release 3153
json 3051, 3055, 3103
JSON 240, 3067, 3071, 3078, 3080
JSON editor 3067, 3071
JSON padding 3067
JSONP 3067

K

keystore 2164
Kibana 2556
Kubernetes 1819, 1820, 1874, 3037, 3038, 3246

L

ldap 1811, 3029
LDAP 1813, 1815, 3031, 3033
Lead Release Manager 3145
lead time 3059, 3170
lead time table 3175
lead-time 3059
lead-time table 3175
Learning 1914
learning resources 2054
link rules 3059, 3080
list view layout 3081
listArtifacts
udclient command 2838
loadEnvironmentCompliancy
udclient command 2840
local 1811, 3029
lockSnapshotConfiguration
udclient command 2841
lockSnapshotVersions
udclient command 2841
lockUser
udclient command 2842
log files 1822, 3040
logging level 1822, 3040
login
udclient command 2842
logout
udclient command 2843
Logstash 2556

M

manage
deployment plan 1836, 3207
task 1845, 3216
manage dashboard 3165
Manage encrypted dataset 976
manage logical environment 3128, 3135
manage pipeline 3128, 3135, 3153
managing authentication resources 233
managing integrations 1874, 1874, 3246, 3246
manual deployments 1840, 3212
manual tasks 1858, 3230
map application environment 3141
mapping 3059
metric 3059
metric bar 3084, 3170
metric definition 3170
metric rules 3145
metrics 3083, 3084, 3091, 3094, 3163, 3164, 3170
metrics data 3083
metrics overview 3083
migrating 1888, 1890, 3260, 3262
milestone risk estimation 3097
moveResource
udclient command 2843
my profile 1820, 3038

N

native integration 1874, 3246
native integrations 1876, 1885, 1888, 3248, 3257, 3260
note tasks 1857, 3229

O

OIDC 1811, 1817, 3029, 3035
open id connect 1811, 1817, 3029, 3035
Open ID Connect 1811, 1817, 3029, 3035
open pipeline deployment 3155
OpenShift 1819, 1820, 3037, 3038
override gate 3153
overview 3163, 3164
Automation 30
home page 1798, 3016
overview of value streams 3049

P

PAC 2926, 2926
application process step
apply-configuration 2934
for-each-agent 2936
for-each-resource-tag 2938
install-component 2940, 2942
join 2945
release-lock 2945
core component process step
add-inventory-status 2964
Core process steps
add-process-warning 2977
application-run-component-process 2930
component-manual-task 2965
component-run-component-process 2975
run-generic-process-for-each-affected-resource 2987
run-operational-process-for-multiple-components 2952
uninstall-component 2957
uninstall-multiple-components 2961
generic process step
acquire-lock 2978
generic-manual-task 2979
plugin 2985
run-generic-process 2986
set-final-process-status 2989
shell 2990
start 2991
switch 2991
remove-inventory-status 2977
rollback-component 2946
rollback-multiple-components 2948
PAC language 2926, 2926
PAC syntax 2926
parser 1874, 1884, 3246, 3256
particles 3049, 3059, 3080, 3103, 3113
password management 1819, 3037
passwords 1819, 3037
pauseWorkflow
udclient command 2845
permissions 1801, 3019
phase 3103
phases 3049
pipeline 1798, 3016, 3128, 3135, 3141, 3145, 3153
pipeline gates 3153
pipelines 3049, 3127
Plotly 3165
plug-in 416
upgrade 416
plugin 2985
plugin defined 1874, 1874, 3246, 3246
plugin SDK 1874, 1874, 1884, 3246, 3246, 3256
plugin types 1876, 1882, 3248, 3254
plugins 1874, 1874, 1875, 1881, 1882, 1884, 1885, 1888, 3078, 3246, 3246, 3247, 3253, 3254, 3256, 3257, 3260
plugins defined 1876, 1882, 3248, 3254
poller 1874, 1884, 3246, 3256
portfolio 3165

preflight builds 371
prepareServerKeystoreImportTask
udclient command 2845
previewUserLockEvent
udclient command 2846
process as code 2926, 2926
process definition 375
process draft 2458
projects 351, 370
create 351, 370
promote deployment plan 1839, 3210
properties 1831, 1832, 1835, 1871, 3203, 3203, 3207, 3243
property examples 1872, 3244
property reference 1872, 3244
provisionEnvironment
udclient command 2846
public key pin 2096, 2164

R

RabbitMQ password 1819, 1820, 1820, 3037, 3038, 3038
Rational Team Concert
Build
2353, 2353
REAT API 1874, 3246
release 1834, 3205
archive 1834, 3205
release details 1831, 1832, 3203, 3203
release edit 1831, 1832, 3203, 3203
release lock 3145
release management 1798, 3016
Release Manager 3145
Release readiness 3186
Release readiness report 3186
release-lock 2945
releases 1798, 1831, 1833, 1871, 3016, 3203, 3205, 3243
Remove
Classification 976
Dataset 976
Encrypt 976
remove-inventory-status 2977
removeAgentFromPool
udclient command 2849
removeAgentFromTeam
udclient command 2850
removeAgentPoolFromTeam
udclient command 2850
removeAgentServerPin
udclient command 2851
removeApplicationFromTeam
udclient command 2852
removeComponentFromApplication
udclient command 2852
removeComponentFromTeam
udclient command 2852
removeComponentResourcePropDef
udclient command 2853
removeComponentTemplateFromTeam
udclient command 2853
removeEnvironmentBaseResource
udclient command 2854
removeEnvironmentFromTeam
udclient command 2855
removeEnvironmentProperty
udclient command 2855
removeGroupFromRoleOnTeam
udclient command 2856
removeInventoryForUnmappedComponents
udclient command 2856
removeProcessFromTeam
udclient command 2857
removeResourceFromTeam
udclient command 2857
removeTagFromAgent
udclient command 2858
removeTagFromApplication
udclient command 2859
removeTagFromComponent
udclient command 2859
removeTagFromResource
udclient command 2860
removeUserFromGroup
udclient command 2860
removeUserFromRoleOnTeam
udclient command 2861
removeVersionFromSnapshot
udclient command 2862
removeVersionLink
udclient command 2862
removeVersionLinkByName
udclient command 2863
removeVersionStatus
udclient command 2863
removeWebhookFromApplication
udclient command 2864
removeWebhookFromProcess
udclient command 2865
repeatApplicationProcessRequest
udclient command 2865
report 1897, 3269
report templates
security report 3193
state of sprint 3186
state of the sprint 3194
reporting 2547
reports 393, 3184
share 1898, 3270
requestApplicationProcess
udclient command 2866
requestProcess
udclient command 2868
reschedule plan 1831, 1832, 3203, 3203
resolving tasks 1841, 1843, 3213, 3215
resources 2054
associating with environments 2345
creating 2313
respondUrl 2388
REST API
agentCLI resource
/teams PUT 1891, 1892, 3263, 3264
agentPools resource 439
/agentPools GET 439
agents resource 439
/{id} DELETE 441
/{id} GET 443
/{id} PUT 445
/agents GET 442
/agents/configureAgents POST 439
/properties GET 442
/properties/agent GET 441
/properties/agent POST 440
/properties/agent PUT 446
/properties/agent/{name} DELETE 440
/properties/environment GET 442
/properties/environment POST 439
/properties/environment PUT 445
/properties/environment/{name} DELETE 440
/properties/locked GET 443
/properties/system GET 444
artifactsets resource 446
/{id} DELETE 446
/{id} GET 447
/{id} PUT 448
/artifactsets GET 447
/artifactsets POST 446
authenticationRealm resource 448
/{realm} DELETE 449
/{realm} GET 448
/{realm} PUT 449
/{realm}/importUsers/{name} PUT 450
/{realm}/moveDown PUT 450
/{realm}/moveUp PUT 449
/{realm}/updateUsers PUT 450
/{realm}/users GET 449
authenticationRealm GET 448
authenticationRealm POST 448
authorizationRealm resource 451
/{realm} DELETE 451
/{realm} GET 451
/{realm} PUT 451
authorizationRealm GET 451
authorizationRealm POST 451
authtoken resource 452
{token} DELETE 452
{token} GET 452
/getOrCreateToken PUT 453
authtoken GET 452
authtoken PUT 452
table GET 452
authtokens resource 453
/{token} DELETE 453
/{token} GET 454
/authtokens GET 453
/authtokens POST 453
build-lives resource 454
/ GET 454
/{id} GET 457, 458, 461, 476, 476
/{name} GET 477, 478
/{path} GET 456, 457, 460, 462
/{set} GET 455, 459
/analytics GET 465, 465
/analyticSummaries GET 466, 466
/artifact-configs GET 455, 459
/artifactRepositories GET 456, 461
/artifacts GET 456, 457, 460, 461
/artifactSets GET 455, 459
/codeCoverage GET 467, 468, 469
/dependencies GET 455, 458
/dependencies POST 455, 458
/dependencies-xml GET 455, 459
/findings GET 471, 472
/issues GET 469, 470
/labels GET 470, 471
/labels POST 462, 462
/latest GET 454
/sourceChange GET 472, 473
/statuses GET 473, 474
/statuses POST 463, 463
/summaries GET 464, 464
/tests GET 474, 475
buildProcessTemplates resource 479
/{id} DELETE 484, 484
/{id} GET 488
/{id} POST 480
/{id} PUT 491
/artifacts GET 486
/artifacts PUT 492
/artifacts/{id} DELETE 485
/buildProcessTemplates GET 489
/buildProcessTemplates POST 483

/copyproperty PUT 482
/graph GET 479
/graph POST 479
/graph PUT 480
/insert POST 480
/iterate DELETE 480
/iterate POST 481
/iterate/property GET 482
/iterate/property POST 481
/iterate/property/{propertyName} DELETE 482
/processes GET 485
/properties GET 487
/properties PUT 492
/property/{name} DELETE 484
/resourceOptions/{id} GET 488
/resources GET 487
/resources PUT 493
/resources/{seq} DELETE 485
/workflow GET 479
codestation-build-lives resource 493
/ GET 494
/{id} DELETE 494
/{id} GET 493
/dependencies GET 495, 495
/latest DELETE 494
/latest GET 493
/properties GET 496, 496
/properties POST 493, 493
/properties PUT 498, 498
/statuses GET 497, 497
codestation-projects resource 498
/{id} DELETE 501, 501
/{id} GET 498, 499
/{id} PUT 507
/build-lives GET 502
/codestation-projects GET 501
/codestation-projects POST 499
/dependencies GET 503, 504
/latest DELETE 501
/latest GET 499
/new/build-lives GET 502
/properties GET 503, 505, 505
/properties POST 500, 500, 500
/properties PUT 507, 508, 508
/statuses GET 506, 506
conf/locale resource 508
/{language}.json GET 508
currentActivity resource 509
/agentDirLocks GET 509
/lockableResources GET 509
/schedules GET 509
dashboard resource 509
/activity GET 509
/activity/{id} GET 510
/projects GET 510
/tags GET 510
DELETE methods
/agents/{id} 441
/agents/{id}/properties/agent/{name} 440
/agents/{id}/properties/environment/{name} 440
/artifactsets/{id} 446
/authtokens/{token} 453
/buildProcessTemplates/{id} 484
/buildProcessTemplates/{id}/artifacts/{id} 485
/buildProcessTemplates/{id}/property/{name} 484
/buildProcessTemplates/{id}/resources/{seq} 485
/buildProcessTemplates/{id}/workflow/graph/{id} 484
/buildProcessTemplates/{id}/workflow/graph/{id}/iterate 480
/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} 482
/codestation-build-lives//{id} 494
/codestation-build-lives//latest 494
/codestation-projects/{id} 501
/codestation-projects/{id}/build-lives/{id} 501
/codestation-projects/{id}/build-lives/latest 501
/fixedAgentPools/{id} 515
/jobPreConditionScripts/{id} 520
/jobs/{id} 522
/jobs/{id}/steps/{id} 521
/plugins{id} 529
/plugins{id}/integrations{id} 529
/processTemplates/{id} 542
/processTemplates/{id}/artifacts/{id} 543
/processTemplates/{id}/property/{name} 542
/processTemplates/{id}/resources/{seq} 543
/processTemplates/{id}/workflow/graph/{id} 542
/processTemplates/{id}/workflow/graph/{id}/iterate 538
/processTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} 540
/projects/{id} 582
/projects/{id}/buildProcesses/{id} 581
/projects/{id}/buildProcesses/{id}/dependencies/{id} 566
/projects/{id}/buildProcesses/{id}/properties/{name} 580
/projects/{id}/buildProcesses/{id}/sourceConfigs/{id} 582
/projects/{id}/buildProcesses/{id}/triggers/{id} 568
/projects/{id}/processes/{id}/properties/{name} 580
/projects/{id}/secondaryProcesses/{id} 582
/projects/{id}/secondaryProcesses/{id}/properties/{name} 581
/projects/{id}/secondaryProcesses/{id}/triggers/{id} 572
/radiator/{id} 624
/reporting/reports/{name} 629
/repositories/{id} 637
/repositories/{id}/trigger 638
/search/buildLives 646
/search/deleteBuild 646
/search/requests 647
/secondaryProcessTemplates/{id} 654
/secondaryProcessTemplates/{id}/artifacts/{id} 654
/secondaryProcessTemplates/{id}/property/{name} 654
/secondaryProcessTemplates/{id}/resources/{seq} 655
/secondaryProcessTemplates/{id}/workflow/graph/{id} 653
/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate 650
/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property/{propertyName} 651
/sourceTemplates/{id} 664
/sourceTemplates/{id}/property/{name} 665
/stage/{id} 670
/statuses/{id} 672
/stepPreConditionScripts/{id} 674
/system/property/{name} 675
/templates/{id} 682
/templates/{id}/property/{name} 682
/workDirScripts/{id} 697
authenticationRealm/{realm} 449
authorizationRealm/{realm} 451
authtoken{token} 452
group/{group} 517
group/{group}/members/{user} 518
notificationSubscriptions/{id} 528
resourceRole/{resourceRole} 640
role/{role} 642
role/{role}/actionMappings 643
team/{team} 676
team/{team}/batchResourceMappings 679
team/{team}/resourceMappings 678
team/{team}/roleMappings 677
user/{user} 692
users/{id} 693
dependencyConfig resource 511
/dependencyConfig GET 511
dynamicAgentPools resource 511
/{id} GET 513
/{id} PUT 511
/dynamicAgentPools GET 512
/ruleConditions GET 513
/ruleTypes GET 513
file resource 514
/{hash} GET 514
files resource 514
/files POST 514
fixedAgentPools resource 514
/{id} DELETE 515
/{id} GET 515
/{id} PUT 516
/fixedAgentPools GET 515
/fixedAgentPools POST 514
GET methods
/agentPools 439
/agents 442
/agents/{id} 443
/agents/{id}/properties 442
/agents/{id}/properties/agent 441
/agents/{id}/properties/environment 442
/agents/{id}/properties/locked 443
/agents/{id}/properties/system 444
/artifactsets 447
/artifactsets/{id} 447
/authtokens 453
/authtokens/{token} 454
/build-lives/ 454
/build-lives//{id} 458
/build-lives//{id}/analytics 465
/build-lives//{id}/analytics/{id} 476
/build-lives//{id}/analytics/{id}/findings 472
/build-lives//{id}/analyticSummaries 466

/build-lives//{id}/analyticSummaries/{name} 478
/build-lives//{id}/analyticSummaries/{name}/summaries 464
/build-lives//{id}/artifact-configs 459
/build-lives//{id}/artifactRepositories 461
/build-lives//{id}/artifactRepositories/{id} 461
/build-lives//{id}/artifactRepositories/{id}/artifacts 461
/build-lives//{id}/artifactRepositories/{id}/artifacts/{path} 462
/build-lives//{id}/artifactSets 459
/build-lives//{id}/artifactSets/{set} 459
/build-lives//{id}/artifactSets/{set}/artifacts 460
/build-lives//{id}/artifactSets/{set}/artifacts/{path} 460
/build-lives//{id}/codeCoverage 469
/build-lives//{id}/dependencies 458
/build-lives//{id}/dependencies-xml 459
/build-lives//{id}/issues 470
/build-lives//{id}/labels 471
/build-lives//{id}/sourceChange 473
/build-lives//{id}/statuses 474
/build-lives//{id}/tests 475
/build-lives//latest 454
/build-lives//latest/analytics 465
/build-lives//latest/analytics/{id} 476
/build-lives//latest/analytics/{id}/findings 471
/build-lives//latest/analyticSummaries 466
/build-lives//latest/analyticSummaries/{name} 477
/build-lives//latest/analyticSummaries/{name}/summaries 464
/build-lives//latest/artifact-configs 455
/build-lives//latest/artifactRepositories 456
/build-lives//latest/artifactRepositories/{id} 457
/build-lives//latest/artifactRepositories/{id}/artifacts 457
/build-lives//latest/artifactRepositories/{id}/artifacts/{path} 457
/build-lives//latest/artifactSets 455
/build-lives//latest/artifactSets/{set} 455
/build-lives//latest/artifactSets/{set}/artifacts 456
/build-lives//latest/artifactSets/{set}/artifacts/{path} 456
/build-lives//latest/codeCoverage 467, 468
/build-lives//latest/dependencies 455
/build-lives//latest/dependencies-xml 455
/build-lives//latest/issues 469
/build-lives//latest/labels 470
/build-lives//latest/sourceChange 472
/build-lives//latest/statuses 473
/build-lives//latest/tests 474
/buildProcessTemplates 489
/buildProcessTemplates/{id} 488
/buildProcessTemplates/{id}/artifacts 486
/buildProcessTemplates/{id}/processes 485
/buildProcessTemplates/{id}/properties 487
/buildProcessTemplates/{id}/resourceOptions/{id} 488
/buildProcessTemplates/{id}/resources 487
/buildProcessTemplates/{id}/workflow 479
/buildProcessTemplates/{id}/workflow/graph 479
/buildProcessTemplates/{id}/workflow/graph/{id}/iterate/property 482
/codestation-build-lives/ 494
/codestation-build-lives//{id} 493
/codestation-build-lives//{id}/dependencies 495
/codestation-build-lives//{id}/properties 496
/codestation-build-lives//{id}/statuses 497
/codestation-build-lives//latest 493
/codestation-build-lives//latest/dependencies 495
/codestation-build-lives//latest/properties 496
/codestation-build-lives//latest/statuses 497
/codestation-projects 501
/codestation-projects/{id} 498
/codestation-projects/{id}/build-lives 502
/codestation-projects/{id}/build-lives/{id} 499
/codestation-projects/{id}/build-lives/{id}/dependencies 504
/codestation-projects/{id}/build-lives/{id}/properties 505
/codestation-projects/{id}/build-lives/{id}/statuses 506
/codestation-projects/{id}/build-lives/latest 499
/codestation-projects/{id}/build-lives/latest/dependencies 503
/codestation-projects/{id}/build-lives/latest/properties 505
/codestation-projects/{id}/build-lives/latest/statuses 506
/codestation-projects/{id}/properties 503
/codestation-projects/new/build-lives 502
/conf/locale/{language}.json 508
/currentActivity/agentDirLocks 509
/currentActivity/lockableResources 509
/currentActivity/schedules 509
/dashboard/activity 509
/dashboard/activity/{id} 510
/dashboard/projects 510
/dashboard/tags 510
/dependencyConfig 511
/dynamicAgentPools 512
/dynamicAgentPools/{id} 513
/dynamicAgentPools/ruleConditions 513
/dynamicAgentPools/ruleTypes 513
/file/{hash} 514
/fixedAgentPools 515
/fixedAgentPools/{id} 515
/jobPreConditions 519
/jobPreConditionScripts 519
/jobPreConditionScripts/{id} 519
/jobs 523
/jobs/{id} 523
/jobs/{id}/processTemplates 522
/jobs/{id}/steps 520
/jobs/{id}/steps/{id} 521
/label/gridSelect 525
/label/itemFileReadStore 525
/myactivity 526
/myactivity/project/{id} 526
/myactivity/workflow/{id} 526
/plugins 530
/plugins{id} 532
/plugins{id}/integrations 529
/plugins{id}/integrations{id} 532
/preflight/build/{buildLifeId} 535
/preflight/request/{requestId} 535
/processRequests{id} 536
/processTemplates 547
/processTemplates/{id} 546
/processTemplates/{id}/artifacts 544
/processTemplates/{id}/processes 543
/processTemplates/{id}/properties 545
/processTemplates/{id}/resourceOptions/{id} 546
/processTemplates/{id}/resources 545
/processTemplates/{id}/workflow 536
/processTemplates/{id}/workflow/graph 537
/processTemplates/{id}/workflow/graph/{id}/iterate/property 539
/projects 551, 604
/projects/{id} 608
/projects/{id}/buildProcesses 601
/projects/{id}/buildProcesses/{id} 606
/projects/{id}/buildProcesses/{id}/activity 615
/projects/{id}/buildProcesses/{id}/activity/{id} 600
/projects/{id}/buildProcesses/{id}/activity/{pageSize}/{page} 614
/projects/{id}/buildProcesses/{id}/buildLives 551
/projects/{id}/buildProcesses/{id}/buildLives/{id} 558
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics 585
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics/{id} 610
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analytics/{id}/findings 594
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries 587
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries/{name} 612
/projects/{id}/buildProcesses/{id}/buildLives/{id}/analyticSummaries/{name}/summaries 583
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifact-configs 559
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories 562
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id} 562
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts 563
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactRepositories/{id}/artifacts/{path} 564

/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets 560
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set} 560
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts 561
/projects/{id}/buildProcesses/{id}/buildLives/{id}/artifactSets/{set}/artifacts/{path} 561
/projects/{id}/buildProcesses/{id}/buildLives/{id}/codeCoverage 590
/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies 558
/projects/{id}/buildProcesses/{id}/buildLives/{id}/dependencies-xml 559
/projects/{id}/buildProcesses/{id}/buildLives/{id}/issues 591
/projects/{id}/buildProcesses/{id}/buildLives/{id}/labels 592
/projects/{id}/buildProcesses/{id}/buildLives/{id}/sourceChange 595
/projects/{id}/buildProcesses/{id}/buildLives/{id}/statuses 597
/projects/{id}/buildProcesses/{id}/buildLives/{id}/tests 599
/projects/{id}/buildProcesses/{id}/buildLives/latest 552
/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics 585
/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics/{id} 609
/projects/{id}/buildProcesses/{id}/buildLives/latest/analytics/{id}/findings 593
/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries 586
/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries/{name} 611
/projects/{id}/buildProcesses/{id}/buildLives/latest/analyticSummaries/{name}/summaries 583
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifact-configs 553
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories 556
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id} 556
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts 556
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactRepositories/{id}/artifacts/{path} 557
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets 554
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set} 554
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts 554
/projects/{id}/buildProcesses/{id}/buildLives/latest/artifactSets/{set}/artifacts/{path} 555
/projects/{id}/buildProcesses/{id}/buildLives/latest/codeCoverage 589
/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies 552
/projects/{id}/buildProcesses/{id}/buildLives/latest/dependencies-xml 553
/projects/{id}/buildProcesses/{id}/buildLives/latest/issues 591
/projects/{id}/buildProcesses/{id}/buildLives/latest/labels 592
/projects/{id}/buildProcesses/{id}/buildLives/latest/sourceChange 595
/projects/{id}/buildProcesses/{id}/buildLives/latest/statuses 597
/projects/{id}/buildProcesses/{id}/buildLives/latest/tests 598
/projects/{id}/buildProcesses/{id}/dashboardActivity 584
/projects/{id}/buildProcesses/{id}/dependencies 564
/projects/{id}/buildProcesses/{id}/dependencies/{id} 566
/projects/{id}/buildProcesses/{id}/dependencies/complex 565
/projects/{id}/buildProcesses/{id}/dependenciesXml 588
/projects/{id}/buildProcesses/{id}/dependencyConfig 588
/projects/{id}/buildProcesses/{id}/dependencySummaries 588
/projects/{id}/buildProcesses/{id}/properties 616
/projects/{id}/buildProcesses/{id}/runtimeProperties 615
/projects/{id}/buildProcesses/{id}/sourceConfigs 596
/projects/{id}/buildProcesses/{id}/sourceConfigs/{id} 613
/projects/{id}/buildProcesses/{id}/statusSummaries 617
/projects/{id}/buildProcesses/{id}/successRateReport 569
/projects/{id}/buildProcesses/{id}/triggers 567
/projects/{id}/buildProcesses/{id}/triggers/{id} 568
/projects/{id}/latestBuilds 573
/projects/{id}/latestStatus 573
/projects/{id}/notificationSubscriptions 572
/projects/{id}/processes 603
/projects/{id}/processes/{id}/properties 616
/projects/{id}/recentProcessActivity 572
/projects/{id}/secondaryProcesses 605
/projects/{id}/secondaryProcesses/{id} 608
/projects/{id}/secondaryProcesses/{id}/dashboardActivity 570
/projects/{id}/secondaryProcesses/{id}/properties 617
/projects/{id}/secondaryProcesses/{id}/triggers 570
/projects/{id}/secondaryProcesses/{id}/triggers/{id} 571
/projects/{id}/sourceChanges 573
/projects/{id}/sourceChanges/changeset/{userName} 600
/projects/{id}/sourceChangesByUser 574
/projects/{id}/sourceChangesByUser/changeset/{userName} 601
/radiator 623
/repo/{id} 624
/repo/{id}/{hash} 624
/repo/{id}/byLabel/{label} 625
/reporting/dimensions 627
/reporting/dimensions/{name} 628
/reporting/dimensions/{name}/expanded 628
/reporting/query/download 629
/reporting/reports 629
/reporting/reports/{name} 630
/reporting/tables 626
/reporting/tables/{name} 626
/reporting/tables/{name}/expanded 627
/reporting/tables/{name}/related 627
/reporting/tablesAndDimensions 628
/reports/analytics/projects/{id} 635
/reports/analytics/projects/{id}/build-processes/{id} 636
/reports/builds/projects 632
/reports/builds/projects/{id} 632
/reports/builds/projects/{id}/build-processes/{id} 632
/reports/commits/projects 632
/reports/commits/projects/{id} 633
/reports/commits/projects/{id}/build-processes/{id} 633
/reports/coverage/projects/{id} 634
/reports/coverage/projects/{id}/build-processes/{id} 635
/reports/issues/projects/{id} 633
/reports/issues/projects/{id}/build-processes/{id} 634
/reports/tests/projects 630
/reports/tests/projects/{id} 631
/reports/tests/projects/{id}/build-processes/{id} 631
/repositories 636
/repositories/{id} 637
/repositories/{id}/trigger 638
/requestStatuses 639
/search/build 647
/search/label 647
/search/projects 645
/search/requests 644
/search/statuses 644
/search/workflows 644
/search/workflows/{id} 645
/secondaryProcessTemplates 659
/secondaryProcessTemplates/{id} 658
/secondaryProcessTemplates/{id}/artifacts 656
/secondaryProcessTemplates/{id}/processes 655
/secondaryProcessTemplates/{id}/properties 656
/secondaryProcessTemplates/{id}/resourceOptions/{id} 657
/secondaryProcessTemplates/{id}/resources 657
/secondaryProcessTemplates/{id}/workflow 648
/secondaryProcessTemplates/{id}/workflow/graph 648
/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property 651
/security/user 662
/sourceTemplates 666
/sourceTemplates/{id} 666
/sourceTemplates/{id}/properties 668

/sourceTemplates/{id}/
sourceConfigs 665
/stage/{id} 670
/stage/{id}/{path:.+} 671
/statuses 671
/statuses/{id} 672
/statuses/nonArchived 673
/stepPreConditionScripts 673
/stepPreConditionScripts/{id} 673
/steps 674
/steps/{id} 674
/system/properties 674
/templates 685
/templates/{id} 683
/templates/{id}/projects 683
/templates/{id}/properties 686
/trends/analytics/{name}/{curr}/
{limit} 690
/trends/analytics/{name}/{curr}/since/
counts/{since} 691
/trends/analytics/{name}/{curr}/since/
severities/{since} 691
/trends/coverages/{name}/{curr} 690
/trends/coverages/{name}/{curr}/since/
{since} 690
/trends/tests/{name}/{curr}/since/
{since} 689
/workDirScripts 696
/workDirScripts/{id} 696
/workDirScripts/{type} 696
/workflow/
massdependencytreenodes 698
/workflow/massdependencytreenodes/
{nodeId} 698
/workflowStatuses 699
authenticationRealm 448
authenticationRealm/{realm} 448
authenticationRealm/{realm}/users 449
authorizationRealm 451
authorizationRealm/{realm} 451
authtoken 452
authtoken{token} 452
authtokentable 452
group 516
group/{group} 517
group/{group}/members 517
notificationSubscriptions 527
notificationSubscriptions/{id} 527
notificationSubscriptions/{id}/
children 528
preferencesavailableLocales 535
resource/{resource}/teamMappings 639
resourceRole 639
resourceRole/{resourceRole} 639
resourceType 640
resourceType/{resourceType} 640
resourceType/{resourceType}/
actions 641
resourceType/{resourceType}/
resourceRoles 641
role 641
role/{role} 642
role/{role}/actionMappings 643
team 675
team/{team} 676
team/{team}/resourceMappings 678
team/{team}/resourceMappings/
{type} 679
team/{team}/roleMappings 676
team/{team}/roleMappings/{role}/
validMembers 677
team/{team}/unmappedResources/
{type} 680
team/{team}/unmappedResources/
{type}/{role} 680
user 691
user/{user} 692
users 693
users/{id} 695
group resource 516
/{group} DELETE 517
/{group} GET 517
/{group} PUT 517
/{group}/members GET 517
/{group}/members/{user} DELETE 518
/{group}/members/{user} PUT 518
group GET 516
group POST 516
import resource 518
/import PUT 518
jobPreConditions resource 518
/jobPreConditions GET 519
jobPreConditionScripts resource 519
/{id} DELETE 520
/{id} GET 519
/{id} PUT 519
/jobPreConditionScripts GET 519
/jobPreConditionScripts POST 519
jobs resource 520
/{id} DELETE 521, 522
/{id} GET 521, 523
/{id} PUT 521, 524
/jobs GET 523
/jobs POST 522
/processTemplates GET 522
/steps GET 520
/steps POST 520
label resource 524
/addToBuild POST 524
/deleteLabel POST 525
/gridSelect GET 525
/itemFileReadStore GET 525
/replaceLabel POST 525
logs resource 525
job/{jobTraceId}/step/{stepSeq}/
{fileName} POST 526
myactivity resource 526
/myactivity GET 526
/project/{id} GET 526
/workflow/{id} GET 526
notificationSubscriptions resource 527
/{id} DELETE 528
/{id} GET 527
/{id} PUT 527
/children GET 528
notificationSubscriptions GET 527
notificationSubscriptions POST 527
plugins resource 528
{id} DELETE 529, 529
{id} GET 532, 532
{id} PUT 534
/allowdowngrade PUT 535
/integrations GET 529
/integrations POST 528
/plugins GET 530
/plugins PUT 534
POST methods
/agents/{id}/properties/agent 440
/agents/{id}/properties/
environment 439
/agents/configureAgents 439
/artifactsets 446
/authtokens 453
/build-lives//{id}/dependencies 458
/build-lives//{id}/labels 462
/build-lives//{id}/statuses 463
/build-lives//latest/dependencies 455
/build-lives//latest/labels 462
/build-lives//latest/statuses 463
/buildProcessTemplates 483
/buildProcessTemplates/{id}/workflow/
graph 479
/buildProcessTemplates/{id}/workflow/
graph/{id} 480
/buildProcessTemplates/{id}/workflow/
graph/{id}/iterate 481
/buildProcessTemplates/{id}/workflow/
graph/{id}/iterate/property 481
/buildProcessTemplates/{id}/workflow/
graph/insert 480
/codestation-build-lives//{id}/
properties 493
/codestation-build-lives//latest/
properties 493
/codestation-projects 499
/codestation-projects/{id}/build-lives/
{id}/properties 500
/codestation-projects/{id}/build-lives/
latest/properties 500
/codestation-projects/{id}/
properties 500
/files 514
/fixedAgentPools 514
/jobPreConditionScripts 519
/jobs 522
/jobs/{id}/steps 520
/label/addToBuild 524
/label/deleteLabel 525
/label/replaceLabel 525
/logsjob/{jobTraceId}/step/{stepSeq}/
{fileName} 526
/plugins{id}/integrations 528
/preflight/lookupArtifactSets 535
/preflight/request 535
/processRequests 536
/processTemplates 541
/processTemplates/{id}/workflow/
graph 537
/processTemplates/{id}/workflow/
graph/{id} 538
/processTemplates/{id}/workflow/
graph/{id}/iterate 539
/processTemplates/{id}/workflow/
graph/{id}/iterate/property 539
/processTemplates/{id}/workflow/
graph/insert 538
/projects 577
/projects/{id}/buildProcesses 575
/projects/{id}/buildProcesses/{id}/
buildLives/{id}/dependencies 559
/projects/{id}/buildProcesses/{id}/
buildLives/{id}/labels 576
/projects/{id}/buildProcesses/{id}/
buildLives/{id}/statuses 579
/projects/{id}/buildProcesses/{id}/
buildLives/latest/dependencies 553
/projects/{id}/buildProcesses/{id}/
buildLives/latest/labels 576
/projects/{id}/buildProcesses/{id}/
buildLives/latest/statuses 579
/projects/{id}/buildProcesses/{id}/
dependencies 565

/projects/{id}/buildProcesses/{id}/sourceConfigs 578
/projects/{id}/buildProcesses/{id}/triggers 567
/projects/{id}/secondaryProcesses 577
/projects/{id}/secondaryProcesses/{id}/triggers 570
/radiator/{id} 623
/repo 624
/repo/{id}/{hash}/copyToServer 625
/repo/{id}/{hash}/copyToUrbanCodeDeploy 626
/repo/{id}/{hash}/label/{label} 625
/reporting/dimensions 628
/reporting/query 628
/reporting/reports 629
/reporting/reports/{name}/output 630
/reporting/tables 626
/repositories 637
/repositories/{id}/trigger 638
/search/deactivateBuild 646
/search/requests 646
/secondaryProcessTemplates 652
/secondaryProcessTemplates/{id}/workflow/graph 649
/secondaryProcessTemplates/{id}/workflow/graph/{id} 650
/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate 650
/secondaryProcessTemplates/{id}/workflow/graph/{id}/iterate/property 651
/secondaryProcessTemplates/{id}/workflow/graph/insert 649
/sourceTemplates 663
/stage 670
/statuses 672
/stepPreConditionScripts 673
/system/properties 675
/templates 681
/workDirScripts 696
/workflow/massdependencytreecopyDependencyAjax 698
/workflow/massdependencytreecreateCodestationDependencyAjax 699
/workflow/massdependencytreecreateWorkflowDependencyAjax 698
/workflow/massdependencytreedeleteDependencyAjax 699
/workflow/massdependencytreemoveDependencyAjax 698
authenticationRealm 448
authorizationRealm 451
group 516
notificationSubscriptions 527
resourceRole 639
role 642
role/{role}/actionMappings 643
team 676
team/{team}/resourceMappings 678
team/{team}/roleMappings 677
user 692
users 694
users/{id}/update 695
preferences resource 535
availableLocales GET 535
preflight resource 535
/build/{buildLifeId} GET 535
/lookupArtifactSets POST 535
/request POST 535
/request/{requestId} GET 535
processRequests resource 536
{id} GET 536
/processRequests POST 536
processTemplates resource 536
/{id} DELETE 542, 542
/{id} GET 546
/{id} POST 538
/{id} PUT 549
/artifacts GET 544
/artifacts PUT 550
/artifacts/{id} DELETE 543
/copyproperty PUT 540
/graph GET 537
/graph POST 537
/graph PUT 537
/insert POST 538
/iterate DELETE 538
/iterate POST 539
/iterate/property GET 539
/iterate/property POST 539
/iterate/property/{propertyName} DELETE 540
/processes GET 543
/processTemplates GET 547
/processTemplates POST 541
/properties GET 545
/properties PUT 550
/property/{name} DELETE 542
/resourceOptions/{id} GET 546
/resources GET 545
/resources PUT 550
/resources/{seq} DELETE 543
/workflow GET 536
projects resource 551, 551
/{id} DELETE 566, 568, 572, 581, 582, 582, 582
/{id} GET 556, 558, 562, 566, 568, 571, 606, 608, 608, 609, 610, 613
/{id} PUT 566, 568, 571, 619, 621, 621, 622
/{name} GET 611, 612
/{path} GET 555, 557, 561, 564
/{set} GET 554, 560
/activity GET 615
/activity/{id} GET 600
/activity/{pageSize}/{page} GET 614
/analytics GET 585, 585
/analyticSummaries GET 586, 587
/artifact-configs GET 553, 559
/artifactRepositories GET 556, 562
/artifacts GET 554, 556, 561, 563
/artifactSets GET 554, 560
/buildLives GET 551
/buildProcesses GET 601
/buildProcesses POST 575
/changeset/{userName} GET 600, 601
/codeCoverage GET 589, 590
/complex GET 565
/dashboardActivity GET 570, 584
/dependencies GET 552, 558, 564
/dependencies POST 553, 559, 565
/dependencies PUT 564
/dependencies-xml GET 553, 559
/dependenciesXml GET 588
/dependencyConfig GET 588
/dependencyConfig PUT 618
/dependencySummaries GET 588
/findings GET 593, 594
/issues GET 591, 591
/labels GET 592, 592
/labels POST 576, 576
/latest GET 552
/latestBuilds GET 573
/latestStatus GET 573
/notificationSubscriptions GET 572
/processes GET 603
/projects GET 604
/projects POST 577
/properties GET 616, 616, 617
/properties PUT 618, 619, 619
/properties/{name} DELETE 580, 580, 581
/recentProcessActivity GET 572
/runtimeProperties GET 615
/secondaryProcesses GET 605
/secondaryProcesses POST 577
/sourceChange GET 595, 595
/sourceChanges GET 573
/sourceChangesByUser GET 574
/sourceConfigs GET 596
/sourceConfigs POST 578
/statuses GET 597, 597
/statuses POST 579, 579
/statusSummaries GET 617
/successRateReport GET 569
/successRateReport PUT 569
/summaries GET 583, 583
/tests GET 598, 599
/triggers GET 567, 570
/triggers POST 567, 570
PUT methods
/agentCLI/teams 1891, 1892, 3263, 3264
/agents/{id} 445
/agents/{id}/properties/agent 446
/agents/{id}/properties/environment 445
/artifactsets/{id} 448
/buildProcessTemplates/{id} 491
/buildProcessTemplates/{id}/artifacts 492
/buildProcessTemplates/{id}/copyproperty 482
/buildProcessTemplates/{id}/properties 492
/buildProcessTemplates/{id}/resources 493
/buildProcessTemplates/{id}/workflow/graph 480
/codestation-build-lives//{id}/properties 498
/codestation-build-lives//latest/properties 498
/codestation-projects/{id} 507
/codestation-projects/{id}/build-lives/{id}/properties 508
/codestation-projects/{id}/build-lives/latest/properties 507
/codestation-projects/{id}/properties 508
/dynamicAgentPools/{id} 511
/fixedAgentPools/{id} 516
/import 518
/jobPreConditionScripts/{id} 519
/jobs/{id} 524
/jobs/{id}/steps/{id} 521
/plugins 534
/plugins{id}/integrations{id} 534
/plugins/allowdowngrade 535
/processTemplates/{id} 549
/processTemplates/{id}/artifacts 550

/processTemplates/{id}/
copyproperty 540
/processTemplates/{id}/properties 550
/processTemplates/{id}/resources 550
/processTemplates/{id}/workflow/
graph 537
/projects/{id} 621
/projects/{id}/buildProcesses/{id} 619
/projects/{id}/buildProcesses/{id}/
dependencies 564
/projects/{id}/buildProcesses/{id}/
dependencies/{id} 566
/projects/{id}/buildProcesses/{id}/
dependencyConfig 618
/projects/{id}/buildProcesses/{id}/
properties 619
/projects/{id}/buildProcesses/{id}/
sourceConfigs/{id} 622
/projects/{id}/buildProcesses/{id}/
successRateReport 569
/projects/{id}/buildProcesses/{id}/
triggers/{id} 568
/projects/{id}/processes/{id}/
properties 618
/projects/{id}/secondaryProcesses/
{id} 621
/projects/{id}/secondaryProcesses/{id}/
properties 619
/projects/{id}/secondaryProcesses/{id}/
triggers/{id} 571
/reporting/reports/{name} 629
/reporting/tables/{name} 627
/repositories/{id} 637
/secondaryProcessTemplates/{id} 660
/secondaryProcessTemplates/{id}/
artifacts 662
/secondaryProcessTemplates/{id}/
copyproperty 652
/secondaryProcessTemplates/{id}/
properties 662
/secondaryProcessTemplates/{id}/
resources 662
/secondaryProcessTemplates/{id}/
workflow/graph 649
/sourceTemplates/{id} 668
/sourceTemplates/{id}/
copyproperty 663
/sourceTemplates/{id}/properties 669
/stage/{id} 670
/stage/{id}/{path:.+} 671
/statuses/{id} 672
/stepPreConditionScripts/{id} 673
/system/properties 675
/templates/{id} 687
/templates/{id}/copyproperty 681
/templates/{id}/properties 688
/workDirScripts/{id} 697
/workflow/{workflowId}/
dependencies/ 697
authenticationRealm/{realm} 449
authenticationRealm/{realm}/
importUsers/{name} 450
authenticationRealm/{realm}/
moveDown 450
authenticationRealm/{realm}/
moveUp 449
authenticationRealm/{realm}/
updateUsers 450
authorizationRealm/{realm} 451
authtoken 452
authtoken/getOrCreateToken 453
group/{group} 517
group/{group}/members/{user} 518
notificationSubscriptions/{id} 527
resourceRole/{resourceRole} 640
role/{role} 642
team/{team} 676
team/{team}/
batchResourceMappings 679
user/{user} 692
user/{user}/setPassword 693
user/{user}/unlock 693
users/{id}activate 694
users/{id}inactivate 694
radiator resource 623
/{id} DELETE 624
/{id} POST 623
/radiator GET 623
repo resource 624
/{id} GET 624
/{id}/{hash} GET 624
/{id}/{hash}/copyToServer POST 625
/{id}/{hash}/copyToUrbanCodeDeploy
POST 626
/{id}/{hash}/label/{label} POST 625
/{id}/byLabel/{label} GET 625
/repo POST 624
reporting resource 626
/{name} DELETE 629
/{name} GET 626, 628, 630
/{name} PUT 627, 629
/dimensions GET 627
/dimensions POST 628
/download GET 629
/expanded GET 627, 628
/output POST 630
/query POST 628
/related GET 627
/reports GET 629
/reports POST 629
/tables GET 626
/tables POST 626
/tablesAndDimensions GET 628
reports resource 630
/{id} GET 631, 631, 632, 632, 633, 633,
633, 634, 634, 635, 635, 636
/projects GET 630, 632, 632
repositories resource 636
/{id} DELETE 637
/{id} GET 637
/{id} PUT 637
/repositories GET 636
/repositories POST 637
/trigger DELETE 638
/trigger GET 638
/trigger POST 638
requestStatuses resource 638
/requestStatuses GET 639
resource resource 639
/{resource}/teamMappings GET 639
resourceRole resource 639
/{resourceRole} DELETE 640
/{resourceRole} GET 639
/{resourceRole} PUT 640
resourceRole GET 639
resourceRole POST 639
resources
agentPools 439
agents 439
artifactsets 446
authenticationRealm 448
authorizationRealm 451
authtoken 452
authtokens 453
build-lives 454
buildProcessTemplates 479
codestation-build-lives 493
codestation-projects 498
conf/locale 508
currentActivity 509
dashboard 509
dependencyConfig 511
dynamicAgentPools 511
file 514
files 514
fixedAgentPools 514
group 516
import 518
jobPreConditions 518
jobPreConditionScripts 519
jobs 520
label 524
logs 525
myactivity 526
notificationSubscriptions 527
plugins 528
preferences 535
preflight 535
processRequests 536
processTemplates 536
projects 551
radiator 623
repo 624
reporting 626
reports 630
repositories 636
requestStatuses 638
resource 639
resourceRole 639
resourceType 640
role 641
search 643
secondaryProcessTemplates 648
security 662
sourceTemplates 663
stage 670
statuses 671
stepPreConditionScripts 673
steps 674
system 674
team 675
templates 681
trends 689
user 691
users 693
workDirScripts 696
workflow/{workflowId/
dependencies 697
workflow/massdependencytree 698
workflowStatuses 699
resourceType resource 640
/{resourceType} GET 640
/{resourceType}/actions GET 641
/{resourceType}/resourceRoles GET 641
resourceType GET 640
role resource 641
/{role} DELETE 642
/{role} GET 642
/{role} PUT 642
/{role}/actionMappings DELETE 643
/{role}/actionMappings GET 643
/{role}/actionMappings POST 643
role GET 641

role POST 642
search resource 643
/build GET 647
/buildLives DELETE 646
/deactivateBuild POST 646
/deleteBuild DELETE 646
/label GET 647
/projects GET 645
/requests DELETE 647
/requests GET 644
/requests POST 646
/statuses GET 644
/workflows GET 644
/workflows/{id} GET 645
secondaryProcessTemplates resource 648
/{id} DELETE 653, 654
/{id} GET 658
/{id} POST 650
/{id} PUT 660
/artifacts GET 656
/artifacts PUT 662
/artifacts/{id} DELETE 654
/copyproperty PUT 652
/graph GET 648
/graph POST 649
/graph PUT 649
/insert POST 649
/iterate DELETE 650
/iterate POST 650
/iterate/property GET 651
/iterate/property POST 651
/iterate/property/{propertyName} DELETE 651
/processes GET 655
/properties GET 656
/properties PUT 662
/property/{name} DELETE 654
/resourceOptions/{id} GET 657
/resources GET 657
/resources PUT 662
/resources/{seq} DELETE 655
/secondaryProcessTemplates GET 659
/secondaryProcessTemplates POST 652
/workflow GET 648
security resource 662
/user GET 662
sourceTemplates resource 663
/{id} DELETE 664
/{id} GET 666
/{id} PUT 668
/copyproperty PUT 663
/properties GET 668
/properties PUT 669
/property/{name} DELETE 665
/sourceConfigs GET 665
/sourceTemplates GET 666
/sourceTemplates POST 663
stage resource 670
/{id} DELETE 670
/{id} GET 670
/{id} PUT 670
/{id}/{path:.+} GET 671
/{id}/{path:.+} PUT 671
/stage POST 670
statuses resource 671
/{id} DELETE 672
/{id} GET 672
/{id} PUT 672
/nonArchived GET 673
/statuses GET 671
/statuses POST 672
stepPreConditionScripts resource 673
/{id} DELETE 674
/{id} GET 673
/{id} PUT 673
/stepPreConditionScripts GET 673
/stepPreConditionScripts POST 673
steps resource 674
/{id} GET 674
/steps GET 674
system resource 674
/properties GET 674
/properties POST 675
/properties PUT 675
/property/{name} DELETE 675
team resource 675
/{team} DELETE 676
/{team} GET 676
/{team} PUT 676
/{team}/batchResourceMappings DELETE 679
/{team}/batchResourceMappings PUT 679
/{team}/resourceMappings DELETE 678
/{team}/resourceMappings GET 678
/{team}/resourceMappings POST 678
/{team}/resourceMappings/{type} GET 679
/{team}/roleMappings DELETE 677
/{team}/roleMappings GET 676
/{team}/roleMappings POST 677
/{team}/roleMappings/{role}/validMembers GET 677
/{team}/unmappedResources/{type} GET 680
/{team}/unmappedResources/{type}/{role} GET 680
team GET 675
team POST 676
templates resource 681
/{id} DELETE 682
/{id} GET 683
/{id} PUT 687
/copyproperty PUT 681
/projects GET 683
/properties GET 686
/properties PUT 688
/property/{name} DELETE 682
/templates GET 685
/templates POST 681
trends resource 689
/{name}/{curr} GET 690
/{name}/{curr}/{limit} GET 690
/{since} GET 689, 690
/counts/{since} GET 691
/severities/{since} GET 691
user resource 691
/{user} DELETE 692
/{user} GET 692
/{user} PUT 692
/{user}/setPassword PUT 693
/{user}/unlock PUT 693
user GET 691
user POST 692
users resource 693
/{id} DELETE 693
/{id} GET 695
/update POST 695
activate PUT 694
inactivate PUT 694
users GET 693
users POST 694
workDirScripts resource 696
/{id} DELETE 697
/{id} GET 696
/{id} PUT 697
/{type} GET 696
/workDirScripts GET 696
/workDirScripts POST 696
workflow/{workflowId/dependencies resource 697
/workflow/{workflowId}/dependencies/ PUT 697
workflow/massdependencytree resource 698
copyDependencyAjax POST 698
createCodestationDependencyAjax POST 699
createWorkflowDependencyAjax POST 698
deleteDependencyAjax POST 699
moveDependencyAjax POST 698
nodes GET 698
nodes/{nodeId} GET 698
workflowStatuses resource 699
/workflowStatuses GET 699
REST commands
deploying applications 1998
exporting component 2008
REST-based user interface 240
restartAgent
udclient command 2870
restore 1834, 3206
deployment plans 1834, 3206
releases 1834, 3206
templates 1834, 3206
restoreArtifacts
udclient command 2870
restoreFromRecycleBin
udclient command 2871
resumeWorkflow
udclient command 2871
Revoke
access 976
Classification 976
revokeAuthToken
udclient command 2872
roles 1801, 3019
rollback-component 2946
rollback-multiple-components 2948
rollbackZInventoryEntries
udclient command 2873
rolling deployments 2441
rules 3145
run another plan tasks 1852, 3224
run another plan tasks example 1852, 3224
run pipeline 3153
run-generic-process 2986
running deployments 1840, 3212

S

safe editing 2458
safe install 1875, 3247
saveEnvironmentGate
udclient command 2873
saveEnvironmentTemplateGate
udclient command 2874
savePostProcessingScript
udclient command 2875
searchZInventoryEntries
udclient command 2876
security 1801, 3019
security considerations

overview mmmccxci
security report 3193
ServiceNow integration 1887, 3259
ServiceNow tasks 1859, 1863, 3231, 3235
Set
Current row 973
Dataset 973
set-final-process-status 2989
setAgentJavaHome
udclient command 2876
setAgentKeypairPolicy
udclient command 2877
setAgentLicenseType
udclient command 2877
setAgentProperty
udclient command 2878
setAgentUnavailable
udclient command 2879
setApplicationProperty
udclient command 2879
setComponentEnvironmentProperty
udclient command 2880
setComponentProperty
udclient command 2881
setComponentVersionPropDef
udclient command 2882
setEnvironmentProperty
udclient command 2883
setLoggingConfig
udclient command 2884
setPreferredJavaPackage
udclient command 2885
setResourceProperty
udclient command 2886
setResourceRoleProperty
udclient command 2887
setSystemConfiguration
udclient command 2887
setSystemProperty
udclient command 2891
settings
authentication 1813, 1815, 1817, 1819, 3031, 3033, 3035, 3037
authorization 1815, 1817, 3033, 3035
calendar 1800, 3018
integrations 1885, 1886, 1888, 1890, 3077, 3257, 3258, 3260, 3262
security 1815, 1817, 3033, 3035
user authentication 1811, 1817, 3029, 3035
user authorization 1811, 1812, 3029, 3030
users 1811, 1812, 3029, 3030
setVersionProperty
udclient command 2892
shell 2990
shell script tasks 1859, 3231
shutdownAgent
udclient command 2893
Simple Mail Transfer Protocol 1800, 3017
single sign-on 1817, 3035
SMTP 1800, 3017
sso 1811, 3029
SSO 1817, 3035
SSO server 1817, 3035
stage 3103
stage circle speed 3103
stages 3049
standby-server 341
start 2991
starting deployments manually 1840, 3212
state of the sprint 3194
state of the sprint report 3194
stochastic 3097
Swagger 1874, 3246
swim lanes 3103, 3113
switch 2991
system properties 1872, 3244

T

table metrics 3175
tags 1834, 1851, 3206, 3223
task 1845, 3216
task create 1847, 3219
task dependencies 1848, 3220
task groups 1849, 1850, 3221, 3222
create 1849, 3221
manage 1850, 3222
task progress 1831, 1832, 3203, 3203
task tags 1851, 3223
tasks 1843, 3215
overview 1843, 3215
types 1843, 3215
teams 1801, 3019
roles 1801, 3019
template deployment plan 1839, 3211
templates 383
Test
accessibility 708
disability 708
keyboard 708
test editor overview 792
testAgent
udclient command 2893
tests
datasets 969, 971
editing 792
testVaultAppRole
udclient command 2894
testVaultCertificate
udclient command 2894
testVaultLdapUser
udclient command 2895
trace 1822, 3040
tree editor 3071
troubleshooting and support 2997
tutorials 245, 1914, 1914

U

UAK 1811, 1820, 3029, 3038
udclient commands
addAgentPoolToTeam 2596
addAgentServerPin 2596
addAgentToPool 2597
addAgentToTeam 2598
addApplicationToTeam 2598
addComponentResourcePropDef 2599
addComponentTemplateToTeam 2600
addComponentToApplication 2600
addComponentToTeam 2601
addEnvironmentBaseResource 2601
addEnvironmentProperty 2602
addEnvironmentToTeam 2603
addGroupToTeam 2604
addJavaPackage 2604
addPostProcessingScriptToTeam 2605
addProcessToTeam 2605
addResourceToTeam 2606
addStatusToSnapshot 2607
addTagToAgent 2608
addTagToApplication 2608
addTagToComponent 2609
addTagToResource 2610
addUserToGroup 2611
addUserToTeam 2612
addVersionFiles 2612
addVersionLink 2613
addVersionLinkWithName 2614
addVersionStatus 2615
addVersionToSnapshot 2616
addWebhookToApplication 2616
addWebhookToApplicationWithCredentials 2617
addWebhookToProcess 2618
addWebhookToProcessWithCredentials 2619
addZInventoryEntries 2620
archiveVersion 2620
autoDiscoverResource 2621
bulkAddAgentServerPin 2621
bulkGetAgentJavaHomes 2622
bulkGetPreferredJavaPackageForAgent 2623
bulkInstallAgentJava 2623
bulkRemoveAgentServerPin 2624
bulkSetAgentJavaHome 2624
bulkSetAgentKeypairPolicy 2625
bulkUninstallAgentJava 2626
cancelApplicationPipelineExecution 2626
cancelApplicationProcessRequest 2627
cancelComponentProcessRequest 2627
cancelMaintenanceBlackoutSeries 2627
cancelMaintenanceBlackoutSeriesEvent 2628
cancelProcessRequest 2628
cancelScheduledDeploymentSeries 2629
cancelScheduledDeploymentSeriesEvent 2629
cancelServerKeystoreImportTask 2630
cancelWorkflow 2630
closeTask 2631
completeServerKeystoreImportTask 2631
copyResource 2632
copyVersion 2633
createAgentPool 2635
createApplication 2636
createApplicationFromTemplate 2637
createApplicationPipeline 2638
createApplicationProcess 2639
createApplicationTemplate 2643
createAuthenticationRealm 2645
createAuthorizationRealm 2647
createAuthToken 2644
createBlueprintDesignerIntegration 2649
createComponent 2649
createComponentProcess 2652
createComponentTemplate 2657
createDeploymentTrigger 2657
createEnvironment 2658
createEnvironmentFromTemplate 2660
createEnvironmentTemplate 2661
createGenericProcess 2662
createGroup 2665
createMaintenanceBlackout 2666
createManyVersions 2666
createPostProcessingScript 2668
createResource 2669
createResourceTemplate 2671
createResourceTemplateWithAgentPrototypes 2672
createSnapshot 2673
createSnapshotOfEnvironment 2674
createStatus 2675
createTeam 2676
createUser 2677
createVaultAppRole 2678
createVaultCertificate 2679
createVaultLdapUser 2679
createVaultSecretStore 2680
createVersion 2680
createWebhookTemplate 2681
deleteAgent 2682

deleteAgentPool 2683
deleteAgentProperty 2683
deleteAllDesiredInventoryVersions 2684
deleteApplication 2685
deleteApplicationPipeline 2685
deleteApplicationProcess 2685
deleteApplicationTemplate 2686
deleteAuthenticationRealm 2686
deleteAuthorizationRealm 2687
deleteBlueprintDesignerIntegration 2687
deleteComponent 2688
deleteComponentVersionPropDef 2688
deleteDeploymentTrigger 2689
deleteEnvironment 2689
deleteEnvironmentGate 2690
deleteEnvironmentTemplate 2690
deleteEnvironmentTemplateGate 2691
deleteFrontendServer 2691
deleteGroup 2692
deleteJavaPackage 2692
deleteMaintenanceBlackout 2693
deletePostProcessingScript 2693
deleteRedundantDesiredInventoryVersions 2694
deleteResource 2694
deleteResourceProperty 2695
deleteResourceRoleProperty 2696
deleteSecretStore 2697
deleteServerKeystoreEntry 2697
deleteSnapshot 2697
deleteStatus 2698
deleteTeam 2699
deleteUser 2699
deleteVaultAppRole 2700
deleteVaultCertificate 2700
deleteVaultLdapUser 2701
deleteVersion 2701
deleteWebhookTemplate 2702
downloadJavaPackage 2702
downloadVersionArtifacts 2703
executeApplicationPipeline 2704
exportApplication 2704
exportApplicationProcess 2705
exportComponent 2705
exportComponentProcess 2706
exportComponentTemplate 2706
exportComponentTemplateProcess 2707
exportDetailedGroup 2708
exportGenericProcess 2708
exportGroup 2709
exportSnapshot 2710
finishedImporting 2710
forceCancelWorkflow 2711
forceRenameAgent 2711
forgetAgentCert 2712
getAgent 2712
getAgentJavaHomes 2714
getAgentMaintenanceTask 2714
getAgentMaintenanceTasks 2714
getAgentPool 2715
getAgentPools 2716
getAgentProperty 2717
getAgentPrototypes 2718
getAgents 2718
getAgentStatuses 2718
getAllApplicationProcessesForApplication 2719
getAllComponentProcessRequestCount 2720
getAllComponentsForTemplate 2720
getAllLatestDesiredInventory 2721
getAllProcesses 2721
getAllTeams 2722
getAllUsersInRoleForTeam 2725
getAllUsersWithEmail 2725
getApplication 2726
getApplicationPipeline 2727
getApplicationPipelineExecution 2728
getApplicationPipelineExecutions 2728
getApplicationPipelines 2729
getApplicationProcess 2729
getApplicationProcessExecution 2730
getApplicationProcessRequest 2733
getApplicationProcessRequestProperties 2733
getApplicationProcessRequestStatus 2734
getApplicationProcessRequestVersions 2736
getApplicationProcessUnfilledProperties 2736
getApplicationProperties 2737
getApplicationProperty 2738
getApplications 2740
getApplicationsProcessRequests 2741
getApplicationsProcessRequestsCount 2744
getApplicationTemplate 2739
getApplicationTemplatePropDefs 2739
getApplicationTemplates 2740
getApprovalTasks 2745
getAuthenticationRealm 2748
getAuthenticationRealms 2748
getAuthorizationRealm 2749
getAuthorizationRealms 2749
getAuthToken 2746
getAuthTokens 2746
getAuthTokensForUser 2747
getAvailablePlugins 2749
getBlueprintNodePropertiesTemplate 2750
getBlueprintsInApplication 2750
getComponent 2751
getComponentEnvironmentProperties 2754
getComponentProcess 2755
getComponentProcessRequest 2756
getComponentProcessRequestCount 2757
getComponentProcessRequests 2757
getComponentProperties 2758
getComponentProperty 2759
getComponents 2762
getComponentsInApplication 2764
getComponentTemplatePropDefs 2759
getComponentTemplates 2760
getComponentVersionPropDefs 2760
getComponentVersions 2761
getConfigurableAgentLicenseTypes 2766
getCurrentActivityList 2766
getDatabaseLatency 2766
getDeploymentTrigger 2766
getDeploymentTriggers 2767
getEnvironment 2767
getEnvironmentBaseResources 2769
getEnvironmentCompliancy 2770
getEnvironmentGate 2770
getEnvironmentProperties 2771
getEnvironmentProperty 2772
getEnvironmentResourceTree 2773
getEnvironmentsInApplication 2775
getEnvironmentTemplate 2773
getEnvironmentTemplateGate 2774
getEnvironmentTemplatePropDefs 2774
getExternalApprovalProcess 2776
getExternalApprovalProcesses 2776
getFirstUserWithEmail 2776
getFrontendServers 2777
getHighWatermarks 2777
getInstalledPlugins 2778
getInventoryChangesForSnapshot 2778
getJavaPackage 2778
getJavaPackages 2779
getJavaPackagesForAgent 2779
getLatestDesiredInventory 2780
getLatestSnapshotForEnvironment 2781
getLLMVendorModels 2780
getLLMVendors 2780
getLoggingConfig 2782
getMaintenanceBlackouts 2782
getOpenIDConnectRealms 2783
getOverlappingArtifacts 2783
getPostProcessingScript 2784
getPostProcessingScripts 2784
getPreferredJavaPackageForAgent 2785
getProcessRequest 2785
getProcessRequests 2785
getRelays 2786
getRelayStatuses 2786
getResource 2787
getResourceInventoryEntriesForComponent 2788
getResourceProperties 2789
getResourceProperty 2790
getResourceRoleProperties 2790
getResourceRoleProperty 2791
getResources 2793
getResourcesInRecycleBin 2795
getResourcesWithComponentVersion 2795
getResourceTemplate 2791
getResourceTemplateChildren 2793
getSecretStore 2796
getSecretStores 2797
getServerKeystoreImportTask 2797
getServerKeystoreInfo 2798
getSnapshot 2798
getSnapshotConfiguration 2800
getSnapshotsInApplication 2801
getSnapshotVersions 2800
getStatus 2802
getStatuses 2804
getStatusList 2803
getSystemConfiguration 2805
getSystemProperties 2807
getSystemProperty 2807
getTagsForType 2808
getTagsOnAgent 2808
getTagsOnApplication 2809
getTagsOnComponent 2810
getTagsOnResource 2811
getTasksForUser 2812
getTeam 2813
getUser 2814
getUserByEmail 2815
getUsersThatCanRunDeploymentTriggers 2816
getVaultAppRole 2817
getVaultAppRoles 2817
getVaultCertificate 2817
getVaultCertificates 2818
getVaultLdapUser 2818
getVaultLdapUsers 2819
getVersionId 2819
getVersionLink 2820
getVersionLinks 2821
getVersionProperties 2822
getWebhookTemplate 2823
getWebhookTemplates 2823
getWorkflowTraceLogs 2824
getZOSVersionCumulativeManifest 2824
getZOSVersionDeployManifest 2825
getZOSVersionsWithCommitId 2825
importApplication 2826
importApplicationProcess 2827
importComponent 2828
importComponentProcess 2829

importComponentTemplate 2830
importComponentTemplateProcess 2831
importGenericProcess 2831
importGroup 2832
importLdapUser 2833
importSnapshot 2834
importVersions 2835
installAgent 2836
installAgentJava 2837
installLocalPlugin 2837
installPlugin 2838
listArtifacts 2838
loadEnvironmentCompliancy 2840
lockSnapshotConfiguration 2841
lockSnapshotVersions 2841
lockUser 2842
login 2842
logout 2843
moveResource 2843
pauseWorkflow 2845
prepareServerKeystoreImportTask 2845
previewUserLockEvent 2846
provisionEnvironment 2846
removeAgentFromPool 2849
removeAgentFromTeam 2850
removeAgentPoolFromTeam 2850
removeAgentServerPin 2851
removeApplicationFromTeam 2852
removeComponentFromApplication 2852
removeComponentFromTeam 2852
removeComponentResourcePropDef 2853
removeComponentTemplateFromTeam 2853
removeEnvironmentBaseResource 2854
removeEnvironmentFromTeam 2855
removeEnvironmentProperty 2855
removeGroupFromRoleOnTeam 2856
removeInventoryForUnmappedComponents 2856
removeProcessFromTeam 2857
removeResourceFromTeam 2857
removeTagFromAgent 2858
removeTagFromApplication 2859
removeTagFromComponent 2859
removeTagFromResource 2860
removeUserFromGroup 2860
removeUserFromRoleOnTeam 2861
removeVersionFromSnapshot 2862
removeVersionLink 2862
removeVersionLinkByName 2863
removeVersionStatus 2863
removeWebhookFromApplication 2864
removeWebhookFromProcess 2865
repeatApplicationProcessRequest 2865
requestApplicationProcess 2866
requestProcess 2868
restartAgent 2870
restoreArtifacts 2870
restoreFromRecycleBin 2871
resumeWorkflow 2871
revokeAuthToken 2872
rollbackZInventoryEntries 2873
saveEnvironmentGate 2873
saveEnvironmentTemplateGate 2874
savePostProcessingScript 2875
searchZInventoryEntries 2876
setAgentJavaHome 2876
setAgentKeypairPolicy 2877
setAgentLicenseType 2877
setAgentProperty 2878
setAgentUnavailable 2879
setApplicationProperty 2879
setComponentEnvironmentProperty 2880
setComponentProperty 2881
setComponentVersionPropDef 2882
setEnvironmentProperty 2883
setLoggingConfig 2884
setPreferredJavaPackage 2885
setResourceProperty 2886
setResourceRoleProperty 2887
setSystemConfiguration 2887
setSystemProperty 2891
setVersionProperty 2892
shutdownAgent 2893
testAgent 2893
testVaultAppRole 2894
testVaultCertificate 2894
testVaultLdapUser 2895
uninstallAgentJava 2896
unlockUser 2896
unsetPreferredJavaPackage 2897
updateAgent 2897
updateApplicationPipeline 2898
updateApplicationTemplate 2898
updateApplicationTemplatePropDefs 2900
updateAuthenticationRealm 2901
updateAuthorizationRealm 2902
updateComponent 2904
updateComponentResourcePropDef 2905
updateComponentTemplatePropDefs 2906
updateEnvironment 2907
updateEnvironmentProperty 2908
updateEnvironmentTemplate 2909
updateEnvironmentTemplatePropDefs 2911
updateExternalApprovalProcess 2912
updateMaintenanceBlackout 2912
updateResource 2913
updateResourceTemplate 2915
updateRoleActions 2916
updateTeam 2916
updateUser 2917
updateUsersInAuthRealm 2918
updateVaultAppRole 2918
updateVaultCertificate 2919
updateVaultLdapUser 2919
updateVaultSecretStore 2920
updateWebhookForApplication 2921
updateWebhookForApplicationWithCredentials 2922
updateWebhookForProcess 2923
updateWebhookForProcessWithCredentials 2923
updateWebhookTemplate 2924
upgradeAgent 2925
uninstallAgentJava
udclient command 2896
unit tests 3170
unlockUser
udclient command 2896
unsetPreferredJavaPackage
udclient command 2897
updateAgent
udclient command 2897
updateApplicationPipeline
udclient command 2898
updateApplicationTemplate
udclient command 2898
updateApplicationTemplatePropDefs
udclient command 2900
updateAuthenticationRealm
udclient command 2901
updateAuthorizationRealm
udclient command 2902
updateComponent
udclient command 2904
updateComponentResourcePropDef
udclient command 2905
updateComponentTemplatePropDefs
udclient command 2906
updateEnvironment
udclient command 2907
updateEnvironmentProperty
udclient command 2908
updateEnvironmentTemplate
udclient command 2909
updateEnvironmentTemplatePropDefs
udclient command 2911
updateExternalApprovalProcess
udclient command 2912
updateMaintenanceBlackout
udclient command 2912
updateResource
udclient command 2913
updateResourceTemplate
udclient command 2915
updateRoleActions
udclient command 2916
updateTeam
udclient command 2916
updateUser
udclient command 2917
updateUsersInAuthRealm
udclient command 2918
updateVaultAppRole
udclient command 2918
updateVaultCertificate
udclient command 2919
updateVaultLdapUser
udclient command 2919
updateVaultSecretStore
udclient command 2920
updateWebhookForApplication
udclient command 2921
updateWebhookForApplicationWithCredentials
udclient command 2922
updateWebhookForProcess
udclient command 2923
updateWebhookForProcessWithCredentials
udclient command 2923
updateWebhookTemplate
udclient command 2924
upgradeAgent
udclient command 2925
upload deployment status 3141
UrbanCode Deploy (Legacy) 1890, 3262
UrbanCode Deploy integrations 1885, 3257
UrbanCode Deploy tasks 1868, 3240
user 1811, 3029
user access 81, 1811, 1820, 3029, 3038
user access keys 1881, 3253
user authorization 1812, 3030
user groups 1819, 3037
users 323
using PAC 2926

V

value stream 1881, 3067, 3071, 3097, 3098, 3253
Value stream 3125, 3125
value stream data 3103
value stream filtering 3103
value stream integrations 3077
value stream management 3049, 3080
Value stream metrics 3083, 3084, 3091
Value stream queries 3117
value stream view 3049, 3080, 3117
Value stream visualization 3103

value streams 3113
value_stream_name-vsm.json 3059
View
Dataset 973, 973
Encrypt 973
viewing value streams 3081
VSM 3067, 3071
VSM JSON 3067, 3071
VSM workflows 3103

W

wait for approval tasks 1870, 3242
Web agent 2095, 2096, 2098, 2164
Web agent relay 2098
webhook 1874, 1884, 3246, 3256
WebSocket 2096, 2097, 2098
wipLimit 3059
Work Items tab 2388
wss 2097

Y

yaml
REST commands 2008