HCL DevOps Automation (Automation)

1.0.1 Documentation

April 2025

Special notice

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

Contents

Chapter 1. Release notes for DevOps Automation 2025.03 (1.0.1)

This document includes product description, information about installation instructions along with the contact information of HCL Customer Support. You can also find the known issues identified in this version of HCL DevOps Automation (Automation).

Product description

DevOps Automation is a cloud-based continuous integration platform. Automation is built on modern, cloud-native technologies that enables product teams to plan, code, test, and deploy the applications. Automation provides a holistic view of the progress in the DevOps cycle. For more information about the platform, see Overview of DevOps Automation on page 13.

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 Automation.

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 4.

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.

Known issues

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

Known issues in DevOps Automation 2025.03 (1.0.1)

The known issues are as follows:

ID	Description
NEXUS00001083	When you log in to Automation and open DevOps Measure or DevOps Release, and then try to install a plugin by navigating to **Settings > Integrations > Available > Install**, you might see the following error message: `Error installing plugin` To work around this problem, you can try after some time and install the plugin.
NEXUS00001095	When you log in to Automation and open DevOps Plan from the switcher, and if you click **Let's Go** on the Plan page, you might see the `Authentication failed: CRMDM0904E Token is invalid` error intermittently. To work around this problem, you can log in to Automation again, and then switch to Plan.
NEXUS00001117	When you perform the following steps in Automation, you might see the Home page instead of the new integration page that affects the task of adding a new integration: 1. Navigate to Measure. 2. Click **Create Value Stream**. 3. Enter the name and select the default team. 4. Click **Create and Configure**. 5. Click the **click here to enable a new Integration** link. The Automation Home page is displayed. To work around this problem, in Measure, go to the **Integrations** page to create an integration, and then create a value stream instead of using the wizard to create an integration (as mentioned in the steps above).
NEXUS00001155	When you perform the following steps in Automation, the application opens an empty window with only the left side panel instead of displaying the report data: 1. Open Measure. 2. Navigate to **Insights**. 3. Click any of the available templates. 4. Click **Run report**. 5. Click **Results** after the run report status turns Green. To work around this problem, you can click the share option for the result, copy the link, and then open the link in a new tab to view the report data.
NEXUS00001170	When you perform the following steps in Automation, you might see the login page of Measure or Release instead of the login page of Automation:

ID	Description
	• Log in to Automation. • Navigate to Measure or Release. • Go to **Settings > Integrations > Available**. • Install the Plan plug-in and integrate it. • Go to the **Value streams** page. • Create a value stream by using the Plan integration as an ALM. • Leave the window idle for some time and then return to the window. • Click **Return to Application**. An error is also displayed. To work around this problem, log in to Automation again by using the Automation URL.
NEXUS00001175	When you log in to Automation and navigate to Plan, and if you click the **Help > API Documentation** option in the Plan sidebar, the application opens a new window with redirection to the Automation Home page instead of the documentation page. To work around this problem, modify the URL of the new page to add `\plan` after the host-name. For example, instead of `<hostname>/swagger-ui/index.html`, use `<host-name>/plan/swagger-ui/index.html`.
NEXUS00001186	When you perform the following steps in Automation, you might see the **My Solution** page instead of the Plan page: 1. Navigate to Plan. 2. Add the AI Integration in Plan. 3. Create a work item by using AI. 4. Click the hyperlink generated for the work item. To work around this problem, note down the work item ID and search for it in Plan.

Chapter 2. System Requirements for DevOps Automation 2025.03 (1.0.1)

This document includes information about hardware and software requirements for HCL DevOps Automation (Automation).

Contents

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 Automation (Automation).

The following table lists the requirements to run Automation on the IBM® Red Hat OpenShift platform.

Resource	**Requirement**
Number of worker nodes	4
Processor	4 CPUs
Memory	16 GiB
Disk space	256 GiB
Network	1 Gbps

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

Resource	**Requirement**
Number of worker nodes	4
Processor	4 CPUs
Memory	16 GiB 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 Automation 2025.03 (1.0.1) on page 7

Operating systems and Containers

You can find details about the supported operating systems and containers for HCL DevOps Automation (Automation).

Contents

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 9.0	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)	1.32	The storage class must support Read-WriteOnce (RWO) and ReadWriteMany (RWX).

Related information

System Requirements for DevOps Automation 2025.03 (1.0.1) on page 7

Host prerequisites

You can find the prerequisites that support the operating capabilities for HCL DevOps Automation (Automation).

Contents

Installation

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	129 or later
Microsoft Edge	129 or later
Mozilla Firefox	130 or later

Related information

System Requirements for DevOps Automation 2025.03 (1.0.1) on page 7

Supported software

You can find information about the software supported in HCL DevOps Automation (Automation).

Supported software	Version
HCL DevOps Plan	3.0.3
HCL DevOps Test Hub	11.0.4
HCL DevOps Deploy	8.1.1.0
HCL DevOps Velocity	5.1.3

Note:

Before you begin the installation, you must verify that the hardware and software meet the minimum system requirements. For details about system requirements for each of the individual solutions that are contained in the DevOps Automation platform, see the following information:

- Plan: Hardware, software, and database requirements
- Test: System Requirements

- Deploy: System Requirements
- Measure and Release: System Requirements

Related information

System Requirements for DevOps Automation 2025.03 (1.0.1) on page 7

Chapter 3. Getting Started Guide

This guide provides an overview of HCL DevOps Automation (Automation). You can find the task flows to get you started with Automation. This guide is intended for new users.

Before you can perform the various tasks described in the *Getting Started Guide* and the other guides, you must install Automation. See Installation of DevOps Automation.

Overview of DevOps Automation

HCL DevOps Automation (Automation) is a platform that provides tools to manage the development cycle of your applications effectively. Automation seamlessly integrates planning, coding, testing, release management, deployment, and monitoring operations in the DevOps loop.

Automation provides the following capabilities:

Cloud-based continuous integration and continuous deployment

DevOps Automation is a cloud-based continuous integration platform. Automation is built on modern, cloud-native technologies that enables product teams to plan, code, test, and deploy the applications. Automation provides a holistic view of the progress in the DevOps cycle. Automation is built to install on Kubernetes Service (K8S).

Automation Capabilities

Automation provides solutions to Plan, Code, Control, Test, Deploy, Release, and Measure your DevOps pipelines efficiently. You can use Deploy to define a deployment process that automatically triggers test cases and have those test insights available directly within Velocity. With the Microsoft Azure DevOps integration, you can also define an Azure DevOps pipeline that includes direct execution of tests by using the Test Hub scalable infrastructure. You can use Plan for defect tracking and Code for software source management and version control. Automation provides the following capabilities:

Planning

- Value stream management
- DORA metrics
- Pages
- Portfolio management
- Team planning
- Test generations
- Generation of work item description and work item summary
- Reporting

Source code management

- Remote development
- Source code management

- Git CLI
- Code review workflow
- Activity timeline
- Notifications
- Custom templates
- Wiki pages
- Package registries

Test

- Web-based continuous testing
- Test data authoring
- Unified test management
- Running of tests from the server by using Docker containers
- Connected agents for existing performance agents
- Role-based access and security
- Project home page and overview statistics
- Reporting and resource monitoring service

Build

- Configuration templates
- Centralize security setup, template creation
- Know application dependencies and set up management
- Docker build plug-in
- Git plug-in
- Hundreds of other integrations

Deploy

- Integrations that replace custom scripting
- Scalable distributed automation
- Support for IBM Z systems
- Visual deployment modeling
- Inventory control
- Quality gates and approvals
- Coordination of multi-container deployments
- Rapid migration to the cloud

Release

Release management

Measure

- Value stream management
- Templates
- Portfolio metrics for DevOps
- Multiple integrations with 3rd party products
- Tame pipelines
- Improvisation of auditability
- Strong governance

Role-based access and security

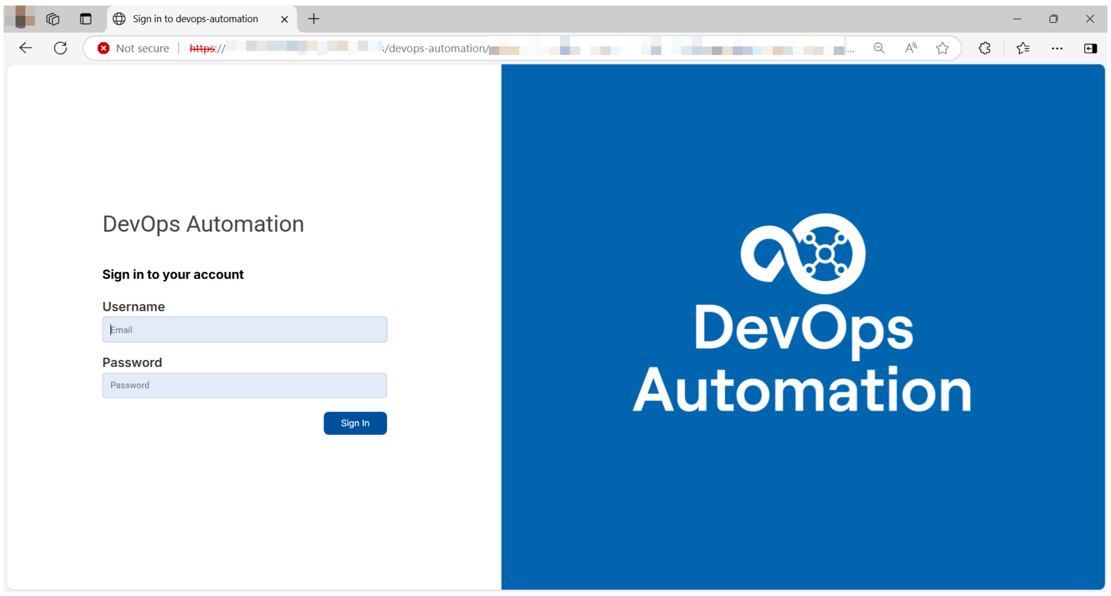
Security is a key concern for IBM clients and therefore, Automation brings a comprehensive, role-based access control scheme to the platform with the admin assigning key permissions (by using roles) for specific members.

DevOps Automation User Interface

You can find an overview of the HCL DevOps Automation (Automation) user interface.

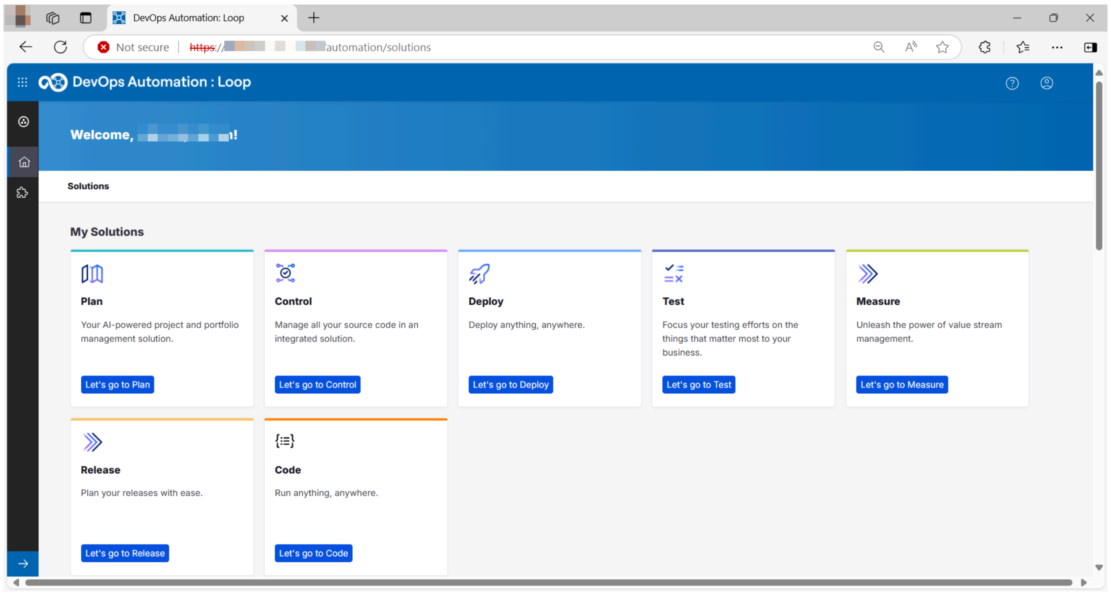
Login page

The login page provides an option to sign in to the DevOps Automation platform.

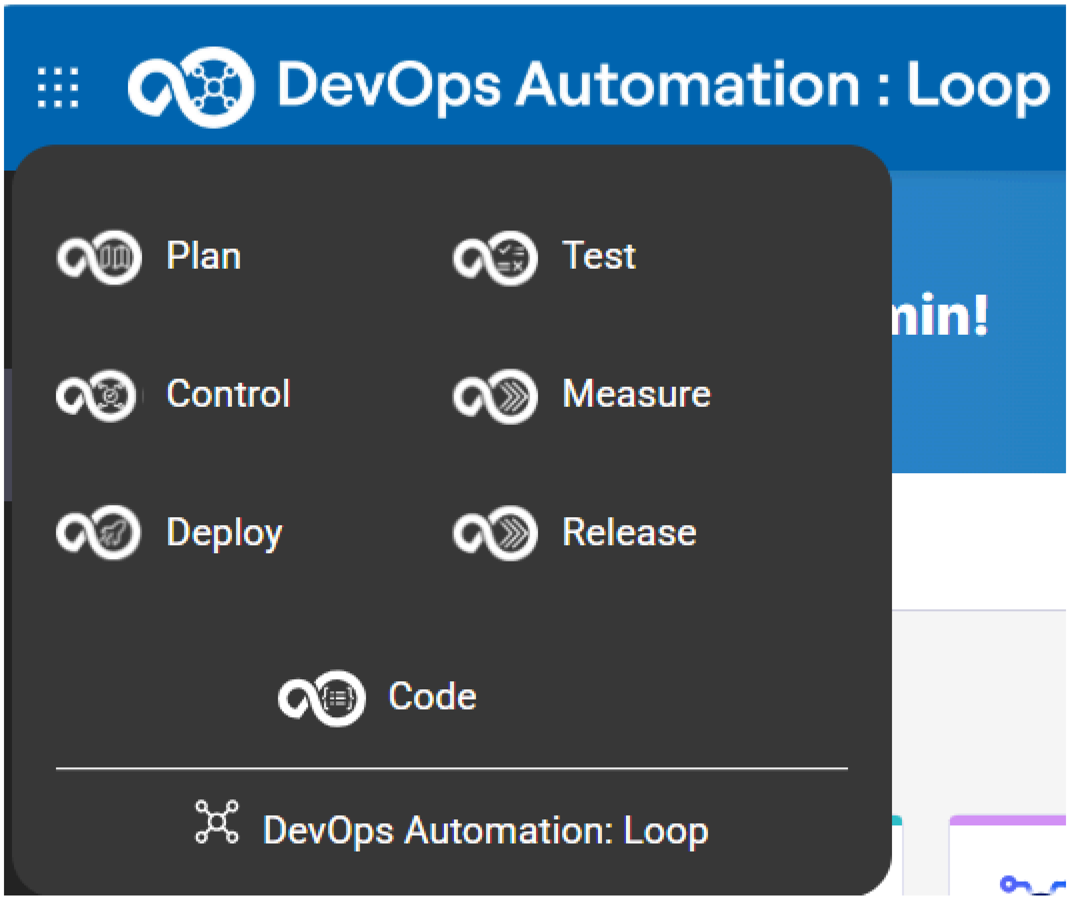


Home page

The home page lists the DevOps tools that you are entitled to use on the DevOps Automation platform.


DevOps Automation: Loop
Not secure | https://[redacted]automation/solutions
DevOps Automation : Loop
Welcome, [redacted]!
Solutions
My Solutions
Plan
Your AI-powered project and portfolio management solution.
Let's go to Plan
Control
Manage all your source code in an integrated solution.
Let's go to Control
Deploy
Deploy anything, anywhere.
Let's go to Deploy
Test
Focus your testing efforts on the things that matter most to your business.
Let's go to Test
Measure
Unleash the power of value stream management.
Let's go to Measure
Release
Plan your releases with ease.
Let's go to Release
Code
Run anything, anywhere.
Let's go to Code

Alternatively, the switcher at the top-left corner of the home page also provides options to switch to a DevOps tool as shown in the following image:



The platform supports the following DevOps tools:

- **Plan**: Provides a sophisticated low-code change management software application designed to facilitate comprehensive change tracking and issue management. It incorporates custom workflow automation, allowing users to tailor processes to their specific needs without extensive coding knowledge.
- **Control**: Provides a source control system with GitHub-compatible APIs.
- **Deploy**: Standardizes and simplifies 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. You can use the tools to model processes that orchestrate complex deployments across every environment and approval gate.
- **Test**: Brings together test data, test environments, and test runs and reports into a single, web-based browser for testers and non-testers.
- **Measure**: Enables you to visualize and measure your DevOps toolchain and data and can better determine the creation of value as work proceeds from idea to customer. Measure provides data that helps teams to identify value creation, bottlenecks, and team issues. Measure consolidates testing and security metrics across your organization. Improves governance across your tools, throughout your organization.
- **Release**: Enables you to standardize and automate processes in your release lifecycle. With Release, you can move releases through your development cycle's phases, from preproduction to production. You can create and implement a predictable schedule of releases for your software applications. If you use team-based planning software, you can integrate related changes to your release plan to make your plan comprehensive and clear. You can openly share your release plan and status with all stakeholders so that they know the schedule and the key milestones and can identify issues that might delay releases.
- **Code**: Enables developers to write, compile, build, and debug code directly in the browser, without local setup. Code is a cloud-based IDE, which is a part of Automation.

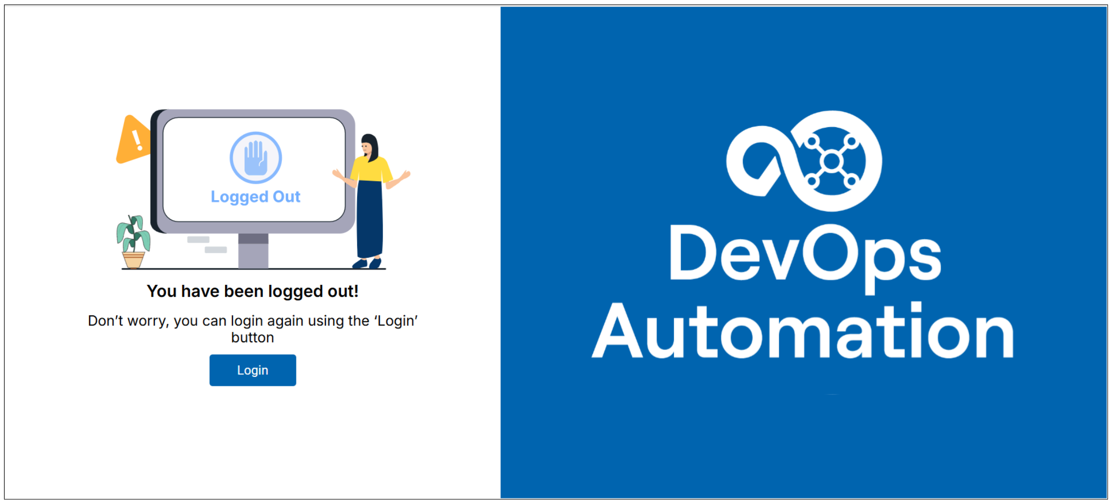
Plugins

The sidebar on the home page provides an option 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.

Logout

To log out of Automation, you can click the profile icon , and then click **Logout**.

Alternatively, you can click the profile icon , and then click **Logout** on any of the solution pages such as Plan, Test, or Deploy. The following LOGOUT page is displayed.


You have been logged out!

Don't worry, you can login again using the 'Login' button

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

Accessibility features of DevOps Automation

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

Automation 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. Administrator Guide

This guide describes how to install the HCL DevOps Automation software, configure licenses, and manage users.

This guide is intended for administrators.

Installation of DevOps Automation

To get started working with HCL DevOps Automation, you must first install Automation.

You can install Automation on the following platforms:

- Kubernetes Service (K8S)

To learn more about the installation of Automation on the supported platforms refer to the following topics:

- Installing on a K8S cluster that has load balancer resources available on page 20
- Installing on a K8S cluster with an upstream L7 load balancer on page 22
- Installing on a K8S cluster that has an upstream L7 load balancer and expects data to be re-encrypted on page 24

Installing DevOps Automation on Kubernetes Service (K8S)

You can find information about the tasks that you can perform to install HCL DevOps Automation (Automation) 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 Automation 2025.03 (1.0.1) on page 7.
- 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 Automation on K8S:

- Installing on a K8S cluster that has load balancer resources available on page 20
- Installing on a K8S cluster with an upstream L7 load balancer on page 22
- Installing on a K8S cluster that has an upstream L7 load balancer and expects data to be re-encrypted on page 24

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 Automation.
3. Make the certificate and key available as a Kubernetes secret in the Kubernetes namespace that you use for Automation.
   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: 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 accelerate-image-secret \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

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

```
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-automation --version 1.0.1
```

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. 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 20 and determined in 5 on page 21 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 accelerate-image-secret \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

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

```
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-automation --version 1.0.1
```

5. 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.

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.
3. Make the certificate and key available as a Kubernetes secret in the Kubernetes namespace that you use for DevOps Automation.
   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 accelerate-image-secret \
    --namespace [namespace_name] \
    --docker-username=<Harbor User ID> \
    --docker-password=<CLI secret> \
    --docker-server=hclcr.io
```

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

```
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-automation --version 1.0.1
```

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 Automation:

```
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.

Management of DevOps Automation features

After you install HCL DevOps Automation (Automation), you can manage certain features based on your requirements.

During the installation of Automation, 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 Automation 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.

Additional Helm parameters

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

As a system administrator, you must complete the additional configuration for the solution 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-prod
- Deploy: hcl-ucd-prod
- Measure: hcl-ucv-prod

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

Configuration of solutions

After installing HCL DevOps Automation (Automation), you must review the configuration required for the individual DevOps solutions.

See the following information on the configuration of the solution:

- Plan: About integration packages, configuration, and testing
- Test: Configuration
- Measure: Configuration
- Deploy: Server settings and configuration

Backup and restoration of the DevOps Automation data

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

A backup is the process of creating a copy of the data on your system and storing it elsewhere, generally on secondary storage. You can then use that copy to recover if your original data is lost or becomes inaccessible.

A restore is a process of copying the backed-up data from the secondary storage and restoring it to its original location. You can restore the backed-up data when your original data is lost or becomes inaccessible.

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 Automation.
- Minimize the downtime of Automation during disaster recovery.

You can find the following information on the backup and restoration of the solution:

- Backup and Recovery of DevOps Velocity
- Backup and restoration of the DevOps Test Hub data
- Backup and recovery of DevOps Deploy

Uninstalling DevOps Automation from Kubernetes Service (K8S)

To reinstall HCL DevOps Automation (Automation) when an ongoing installation fails, you can uninstall Automation and its components from the Kubernetes Service (K8S) cluster.

Before you begin

You must have completed the following tasks:

- Installed Automation.
- Closed Automation, any open web browsers, and all other applications that are enabled by Automation.
- **Optional**: Backed up data from the previous version of Automation.

1. Switch to your K8S cluster by using the following command:

```
kubectl config use-context <context-name>
```

2. Run the following command to uninstall Automation:

   ```
   helm uninstall $HELM_NAME -n $NAMESPACE
   ```

   Where,

   ```
   NAMESPACE=devops-automation
   HELM_NAME=devops-automation
   ```

   The PersistentVolumeClaims and PersistentVolumes that were created during the installation are not deleted automatically. If you reinstall Automation, 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 Automation from the K8S cluster.

DevOps Automation licensing information

You can find the details of licensing that HCL DevOps Automation (Automation) supports.

Automation supports the following 3 tiers of license:

- **Essentials** (Tier 1): DevOps Plan, DevOps Control, DevOps Code
- **Standard** (Tier 2): DevOps Plan, DevOps Control, DevOps Code, DevOps Test, DevOps Deploy
- **Premium** (Tier 3): DevOps Plan, DevOps Control, DevOps Code, DevOps Test, DevOps Deploy, DevOps Measure, DevOps Release

Each tier is defined by the number of DevOps products that it includes on the DevOps Automation platform.

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 Automation, you must specify the License server URL and ID, which are configured only through the Helm chart.

Note: In this release, there is no user interface to view or manage licensing. You can set the licensing information only by using the License server URL and ID details in 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 Automation, 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.

Note: To enable or disable a user's license, contact HCL support. For more information, refer to HCL Customer Support.

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.

User administration

After you install HCL DevOps Automation (Automation), you must consider how the platform manages users and authentication.

You can use the default user management provided by the platform and decide what additional controls you might want to add. If you manage users and authentication through social login, you can review how to use that server to manage Automation users.

Default user administration

HCL DevOps Automation (Automation) 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 Automation.

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

  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 Automation, 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 Automation.

  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 Automation.

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.

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

You can enable social authentication for users to sign up and log in to HCL DevOps Automation (Automation) 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 Automation.
- 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 Automation is not visible when you install Automation 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 Automation.

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 Automation.

What to do next

You must add the self-registered users to the **Users** group in the Keycloak Admin Console. See Default user administration on page 31.

Related information

Keycloak

DevOps Code

DevOps Code (Code) is a cloud-based IDE, which is a part of HCL DevOps Automation (Automation) that allows developers to write, compile, build, and debug code directly in the browser, without local setup.

Built on Visual Studio Code, Code comes with 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.

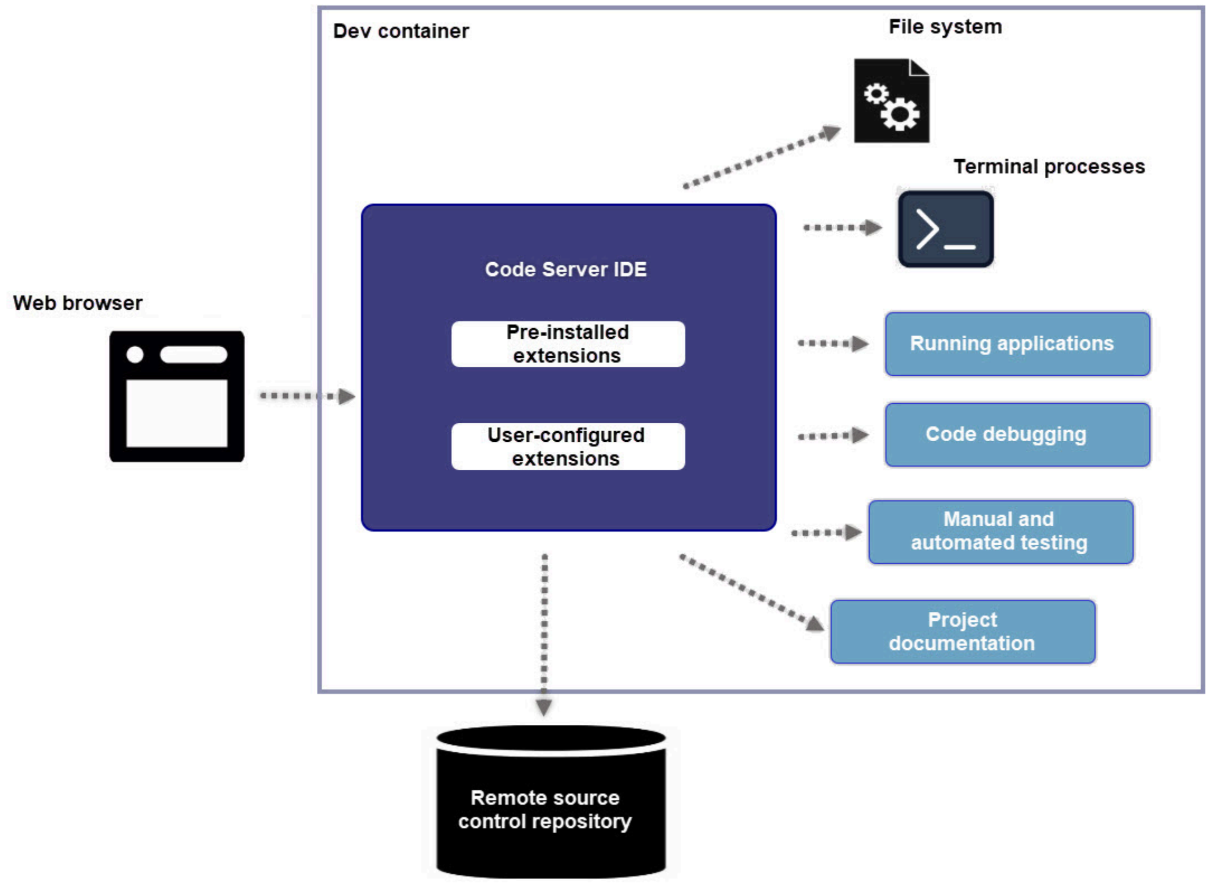
Continue reading to learn how Code enhances your development experience and integrates with Automation.

DevOps Code Overview

DevOps Code (Code) is a cloud-based integrated development environment on HCL DevOps Automation (Automation), 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 of the dev container

The general architecture of the Code platform is shown in the following diagram.



Code provides personal dev containers that are configured for individual users. 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, 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 xxxix. 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. There is built-in support for Git repositories (for example hosted in DevOps Control). Other types of source control repositories are supported by means of extensions that you can install into the dev container.

Getting Started

You can find the basic steps for setting up a workspace in the DevOps Code (Code) Browser IDE.

You can find instructions on how to perform the following tasks:

- Clone a remote Git repository.
- Run a source file from the terminal.
- Set up port forwarding to access your application from your web browser.

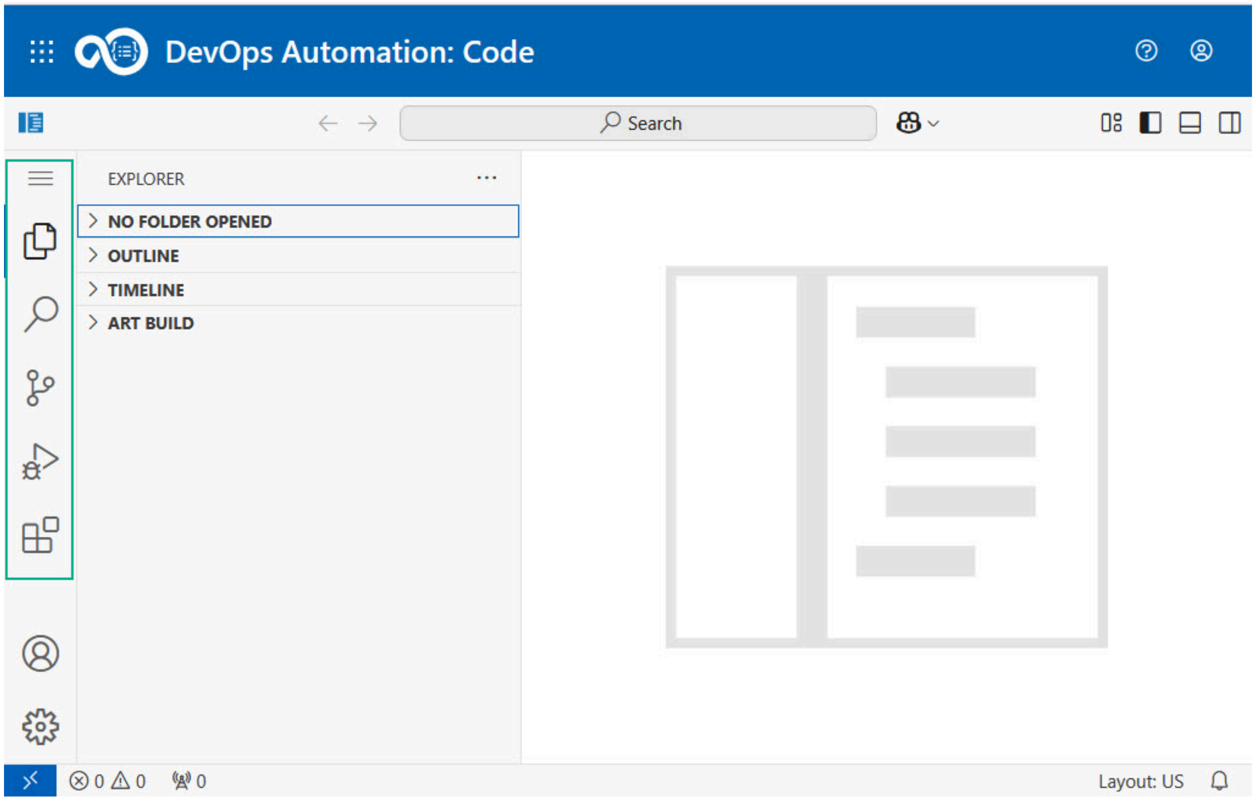
For a visual walkthrough, watch the following video. By following the steps, you can quickly get your project running in the browser IDE.

../videos/hello-world-qs-devcode.mp4

User Interface

The DevOps Code (Code) 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 list outlines the main highlights of the Code user interface:

Menu item	Description
Main 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.
Folder button	Opens the Explorer panel that provides tree-like navigation of the workspace structure.
Search button	Provides the search and replace functionality for files in the workspace.
Source control button	Provides the source control panel from where you can perform commands on Git repositories that are used in your workspace.
Run and Debug button	Provides common actions for running and debugging applications.
Extensions button	Provides a menu to manage extensions.

For more information about the UI of the browser IDE, refer to the documentation of Visual Studio Code.

Management of Dev Containers

With DevOps Code (Code), you can set up the initial process and experience the faster loading of preconfigured containers on subsequent logins. When you terminate a dev container, all files and unsaved work are deleted. Therefore, you must push changes to remote source control before termination.

When you access Code for the first time from the landing page of HCL DevOps Automation (Automation), it takes some time to launch the dev container.

During the initial setup, necessary dependencies, extensions, and libraries are installed and configured to create a standard environment (Java and C++). On subsequent logins, Code loads your pre-configured dev container and workspace much faster.

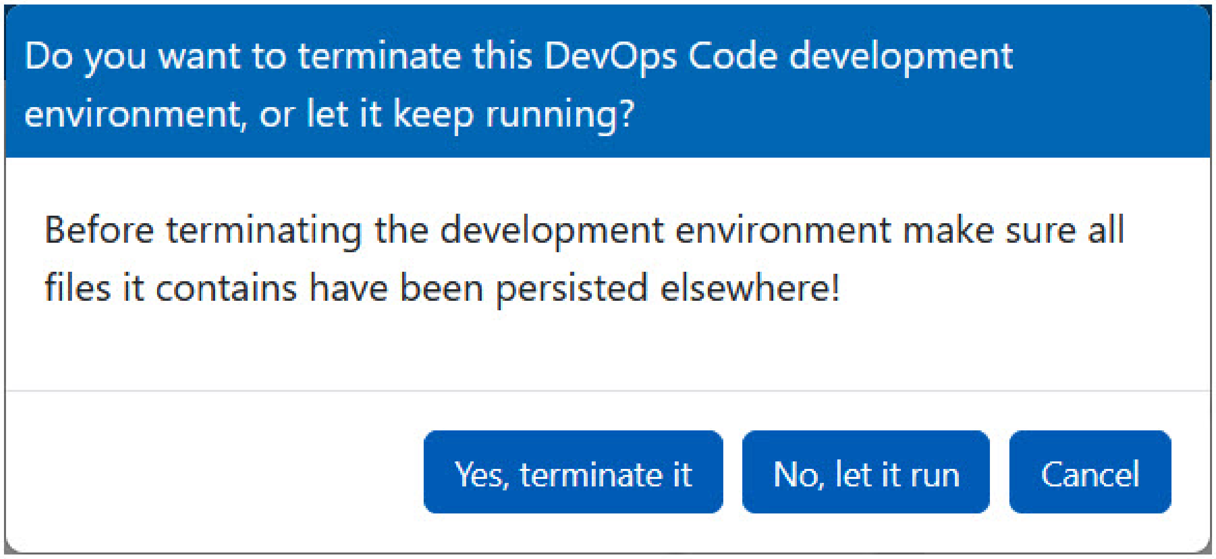
Terminating a dev container

Code manages dev containers by preserving your working environment. The browser IDE settings are saved within the dev container file system. For more information on user and workspace settings, refer to settings.

Note: When you sign out and choose to terminate the dev container, all files in the dev container, including the IDE settings, are lost.

Warning: When you terminate the dev container, all processes within the dev container will come to a halt. You must ensure to push all changes to the remote source control repository before doing so, as terminating the dev container deletes the file system, including workspace folders and any unsaved files, clearing your current workspace and discarding any unsaved work (See the general architecture diagram on page xxxv).

If you prefer to continue your work, you can let the dev container keep running when you logout. You can then return to your dev container later and find it in the exact same state as when you logged out. However, if you do not plan to use it soon, you should terminate the container to free up resources.

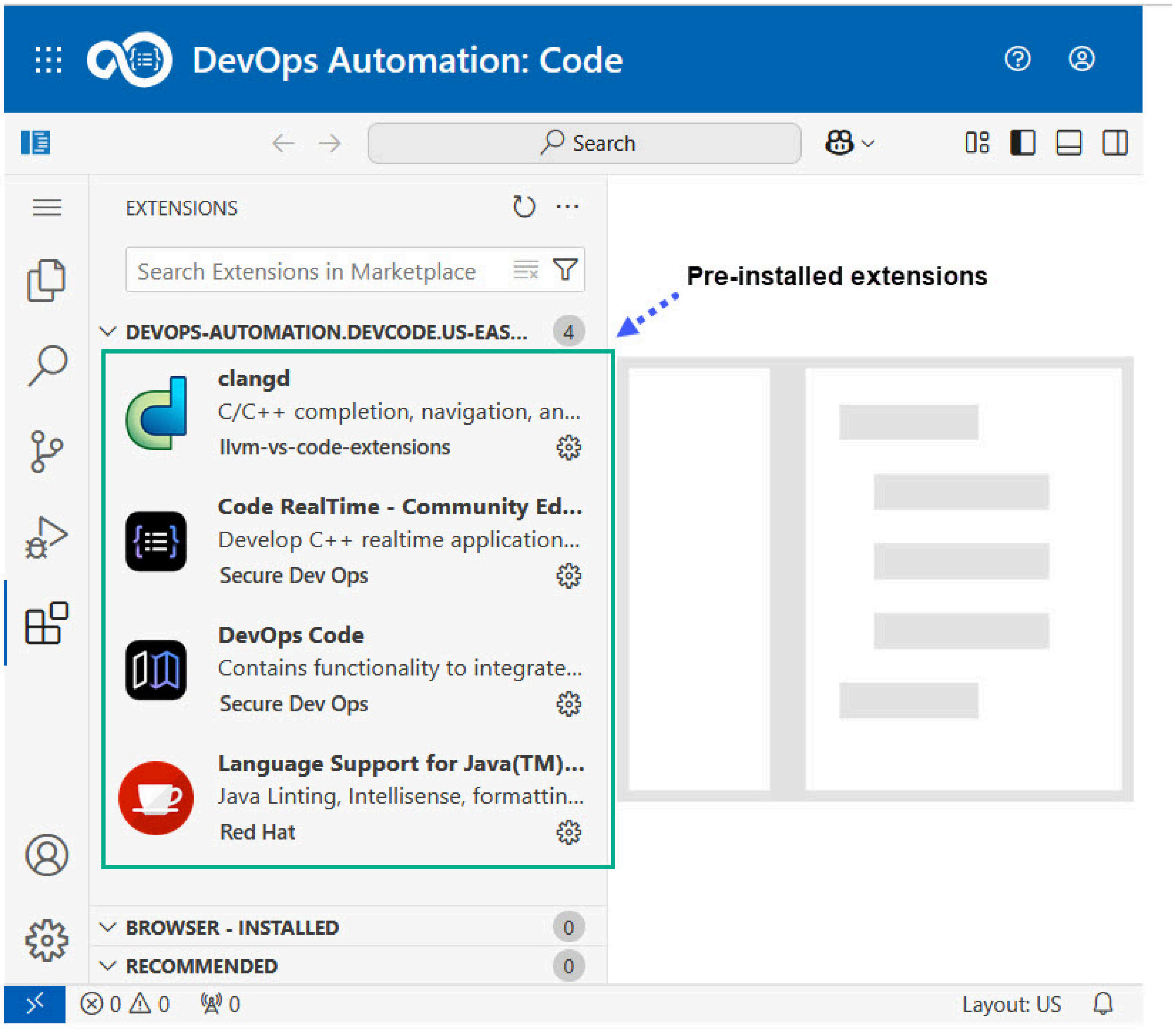

Do you want to terminate this DevOps Code development environment, or let it keep running?

Before terminating the development environment make sure all files it contains have been persisted elsewhere!

Extensions

DevOps Code (Code) includes extensions such as DevOps Code for change management, DevOps Code RealTime for event-driven C++ development, Clangd for C++ support, and Red Hat Java Support. These tools enhance coding, debugging, and version control, with options for additional extensions.

The following image shows the preinstalled extensions available in Code.


DevOps Automation: Code
Search
EXTENSIONS
Search Extensions in Marketplace
Pre-installed extensions
DEVOPS-AUTOMATION.DEVCODE.US-EAS... 4
clangd
C/C++ completion, navigation, an...
llvm-vs-code-extensions
Code RealTime - Community Ed...
Develop C++ realtime application...
Secure Dev Ops
DevOps Code
Contains functionality to integrate...
Secure Dev Ops
Language Support for Java(TM)...
Java Linting, Intellisense, formattin...
Red Hat
BROWSER - INSTALLED 0
RECOMMENDED 0
0 0 0
Layout: US

Code offers a range of preinstalled extensions from the Open VSX Registry, designed to enhance the development experience:

- DevOps Code: This extension lets you create work items in HCL DevOps Plan (Plan), for example when you encounter defects while debugging your application. The **DevOps Code** extension allows integration with Plan, enabling you to create work items directly from your development environment.
- DevOps Code RealTime: This extension supports the development of event-driven, stateful, applications in C++ using the Art language.
- Clangd C++ Language Support: This extension supports C++ development, the clangd plugin. It enables a more efficient and error-free development process for C++ developers.
- Red Hat Language Support for Java: This extension offers comprehensive language support for Java, making it easier to build robust and scalable Java applications.

These preinstalled extensions enhance your development environment by providing tools for coding, debugging, version control, and application development that are all within the same IDE. You can install additional extensions from the Open VSX registry. You can also install local VSIX files in the dev container. However, any extensions you install will be removed when you terminate the dev container. See Management of Dev Containers in Code on page xxxviii.

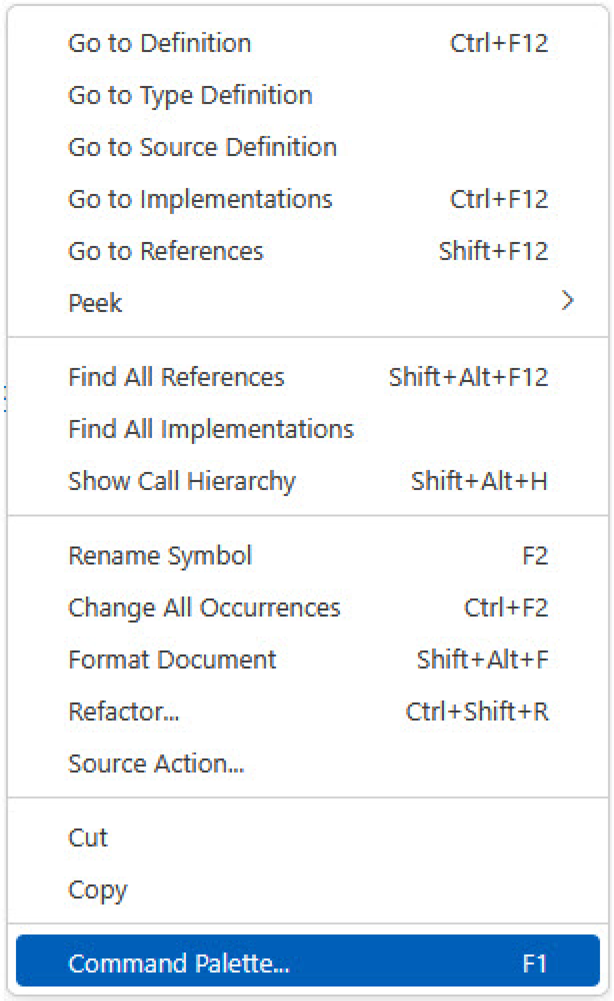
Creating Plan Items with DevOps Code

The DevOps Code extension allows integration with HCL DevOps Automation (Automation), enabling you to create and manage work items directly from your development environment in the browser IDE.

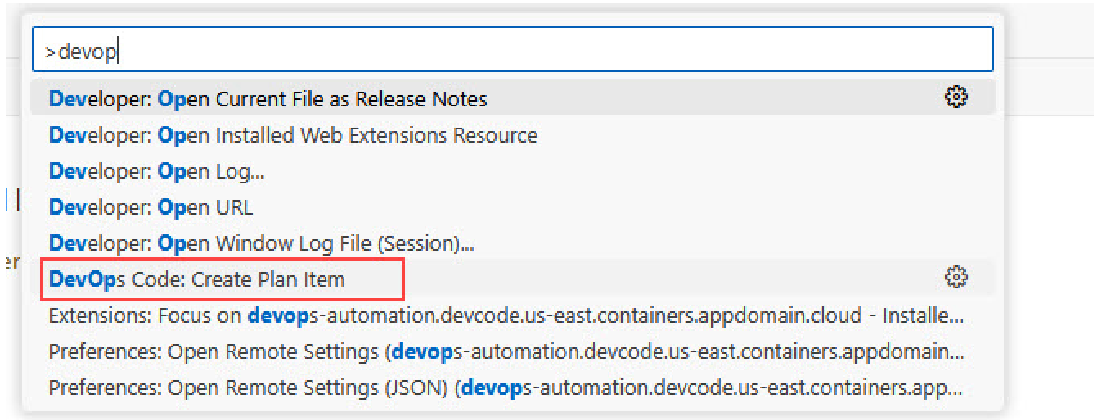
About this task

Follow these steps to configure DevOps Code extension in the DevOps Code (Code) browser IDE, and create work items in HCL DevOps Plan (Plan).

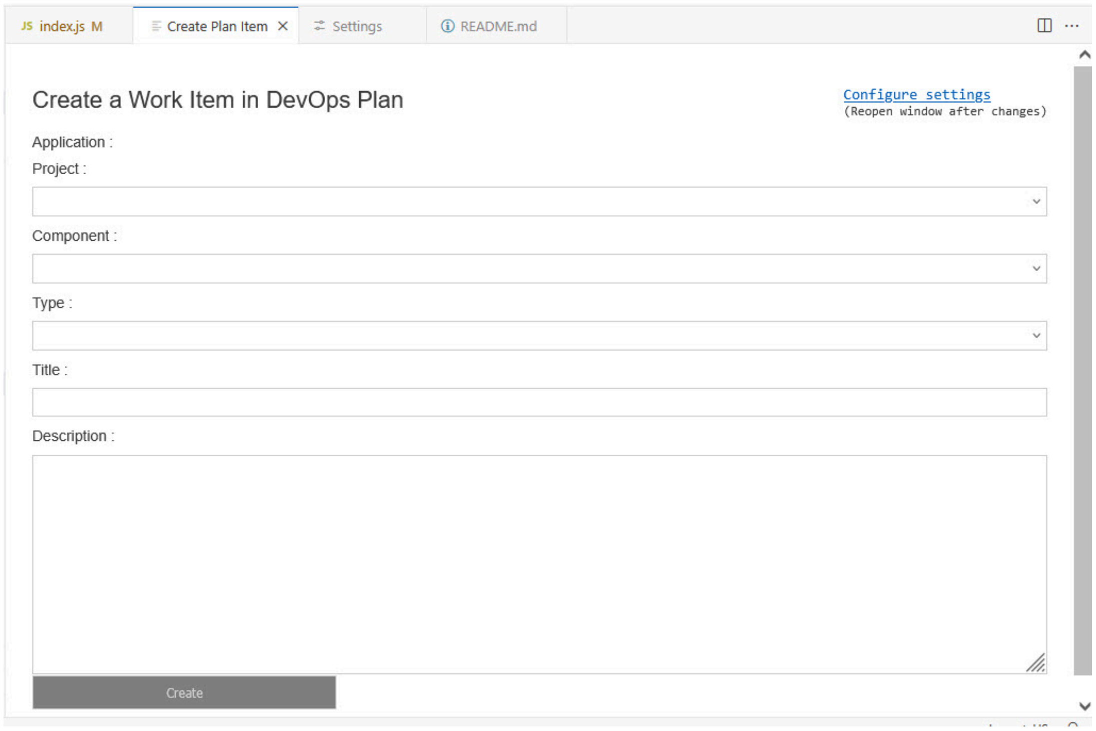
1. Open the Command Palette.
   Right-click to open the context menu and select **Command Palette** in the editor.


Go to Definition Ctrl+F12
Go to Type Definition
Go to Source Definition
Go to Implementations Ctrl+F12
Go to References Shift+F12
Peek
Find All References Shift+Alt+F12
Find All Implementations
Show Call Hierarchy Shift+Alt+H
Rename Symbol F2
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Refactor... Ctrl+Shift+R
Source Action...
Cut
Copy
Command Palette... F1

2. Access the DevOps Code Command.
    a. Choose the **DevOps Code** command from the list of available commands.


>devop
Developer: Open Current File as Release Notes
Developer: Open Installed Web Extensions Resource
Developer: Open Log...
Developer: Open URL
Developer: Open Window Log File (Session)...
DevOps Code: Create Plan Item
Extensions: Focus on devops-automation.devcode.us-east.containers.appdomain.cloud - Installe...
Preferences: Open Remote Settings (devops-automation.devcode.us-east.containers.appdomain...
Preferences: Open Remote Settings (JSON) (devops-automation.devcode.us-east.containers.app...

The **Create Plan Item** window is displayed.



Create a Work Item in DevOps Plan

Configure settings
(Reopen window after changes)

Application :

Project :

Component :

Type :

Title :

Description :

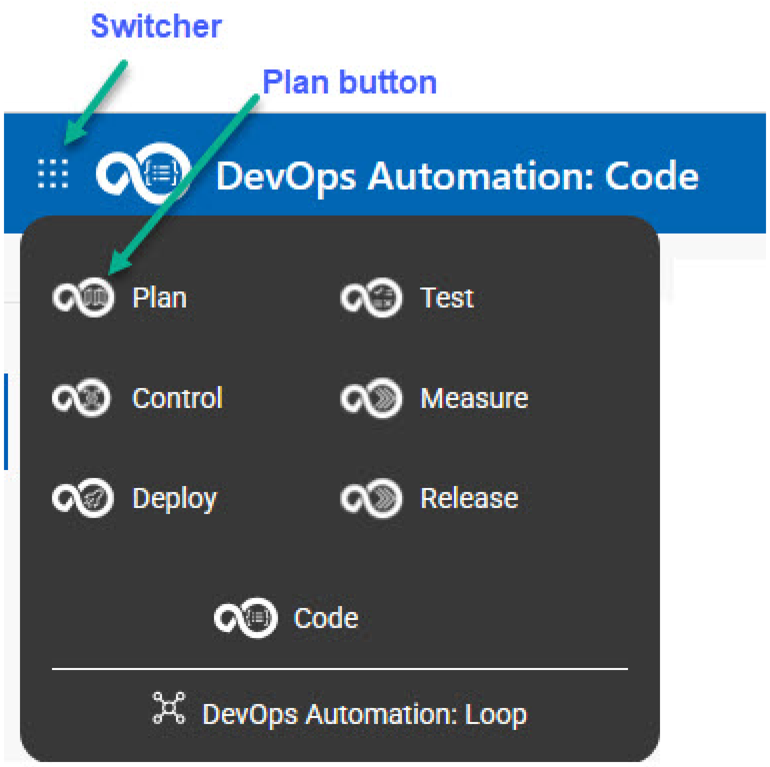
b. Click the **Configure settings** button to integrate DevOps Code with Plan for the first time.

   The **Settings** window is displayed to add the Personal Access Token, Plan Server URL, and Team Space ID.

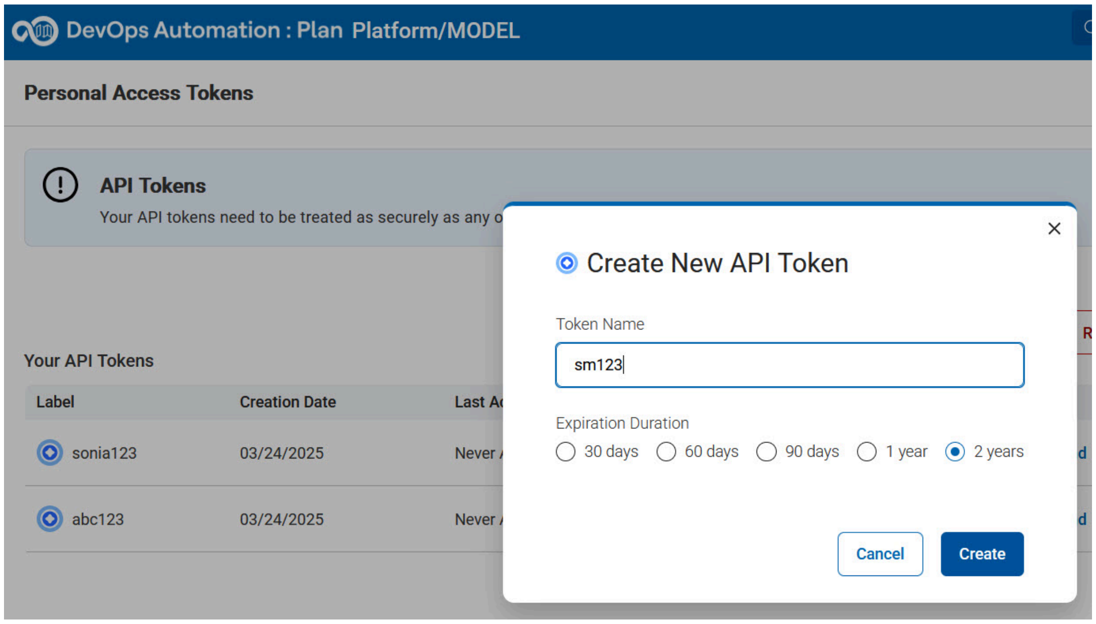
   You can add these fields by navigating to the Plan platform, which is detailed in the next step.

3. Perform the following steps to get configuration details from Plan:

a. Click the **Plan** button in the **DevOps Automation** 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.



Personal Access Tokens

API Tokens
Your API tokens need to be treated as securely as any o

Your API Tokens

Label	Creation Date	Last A
sonia123	03/24/2025	Never
abc123	03/24/2025	Never

Create New API Token

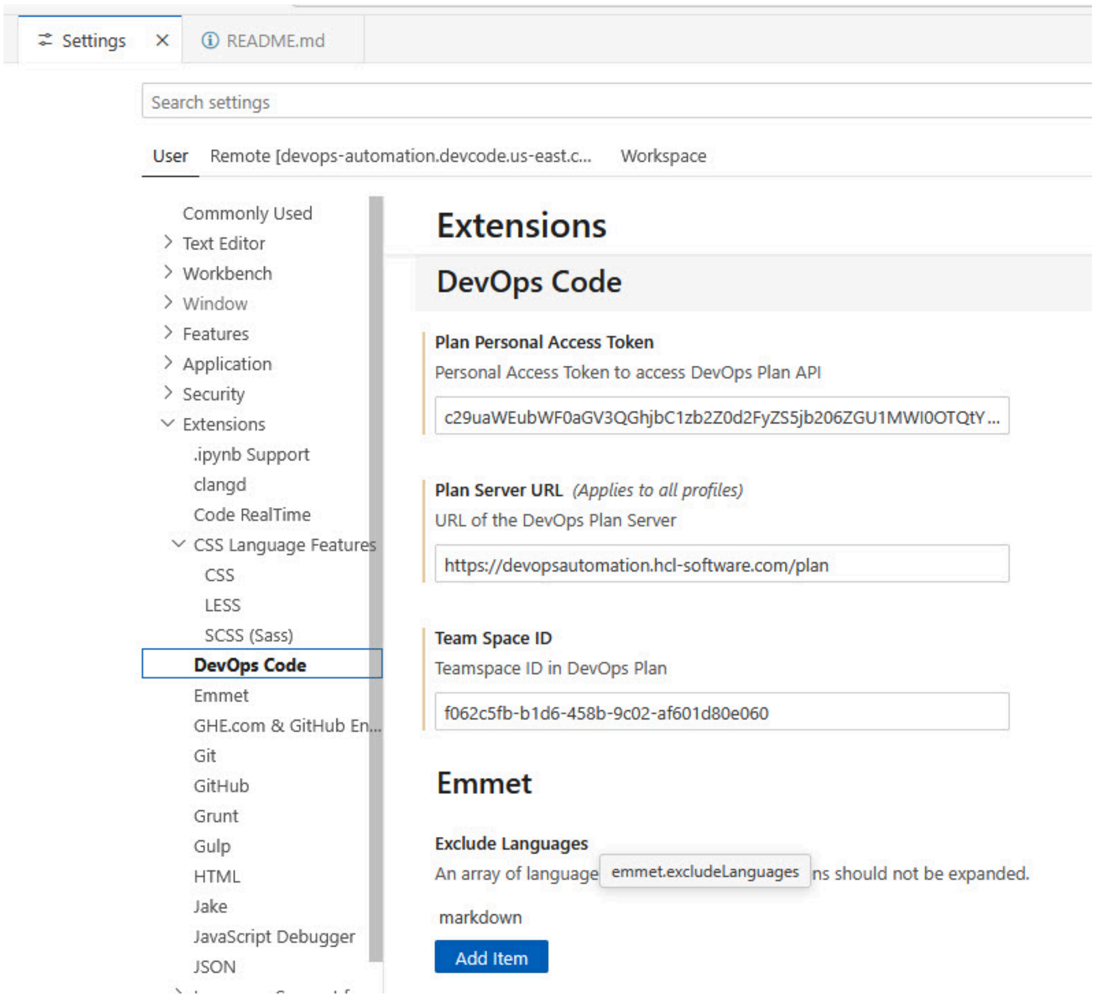
Token Name: sm123

Expiration Duration: 30 days 60 days 90 days 1 year [✓ 2 years]

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://devopsautomation.hcl-software.com/plan/#/ccmweb/view/repos/f062c5fb-b1d6-458b-9c02-af601d80e060/databases/MODEL/records/WorkItem/MODEL00000179



Extensions

DevOps Code

Plan Personal Access Token
Personal Access Token to access DevOps Plan API
c29uaWEubWF0aGV3QGhjbC1zb2Z0d2FyZS5jb206ZGU1MWI0OTQtY...

Plan Server URL *(Applies to all profiles)*
URL of the DevOps Plan Server
https://devopsautomation.hcl-software.com/plan

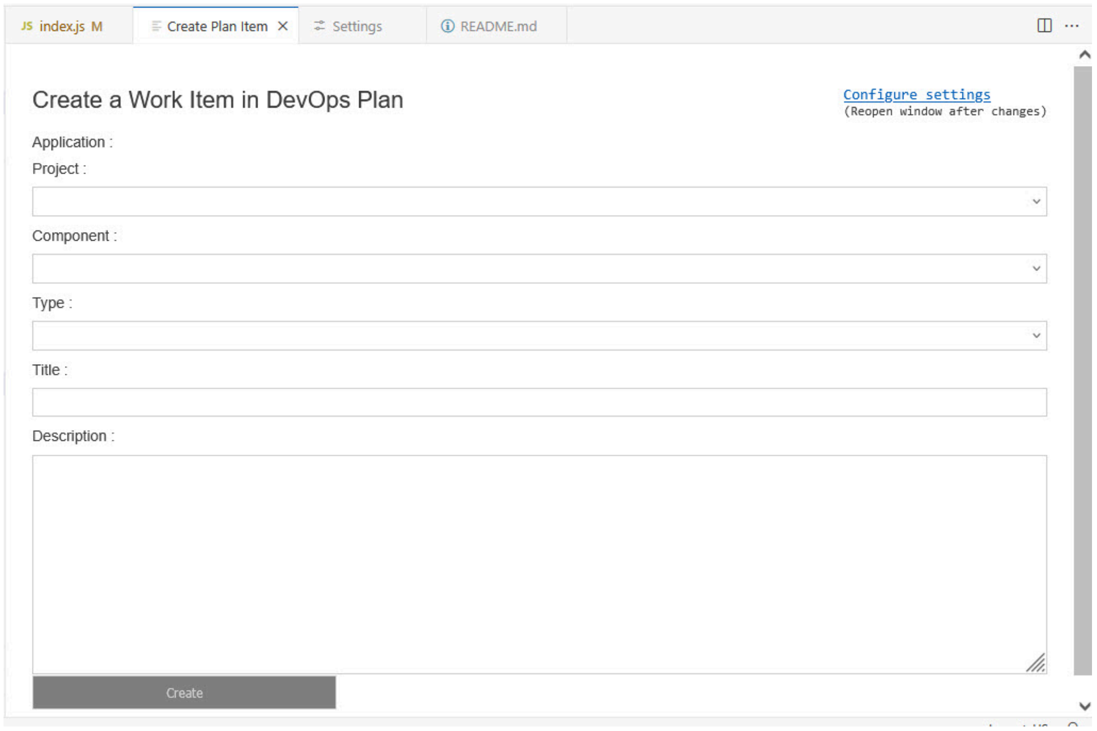
Team Space ID
Teamspace ID in DevOps Plan
f062c5fb-b1d6-458b-9c02-af601d80e060

Emmet

Exclude Languages
An array of language emmet.excludeLanguages ns should not be expanded.
markdown

5. Create a work item:
   a. Select the **DevOps Code** command from **Command Palette**.

      The **Create Plan Item** window is displayed again.



Create a Work Item in DevOps Plan

Configure settings
(Reopen window after changes)

Application :

Project :

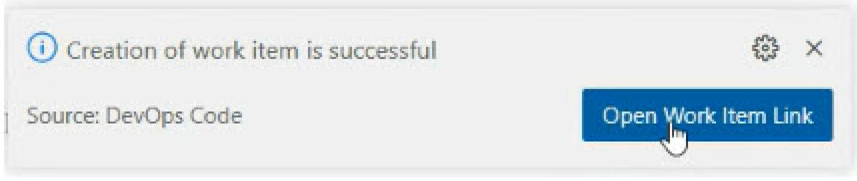
Component :

Type :

Title :

Description :

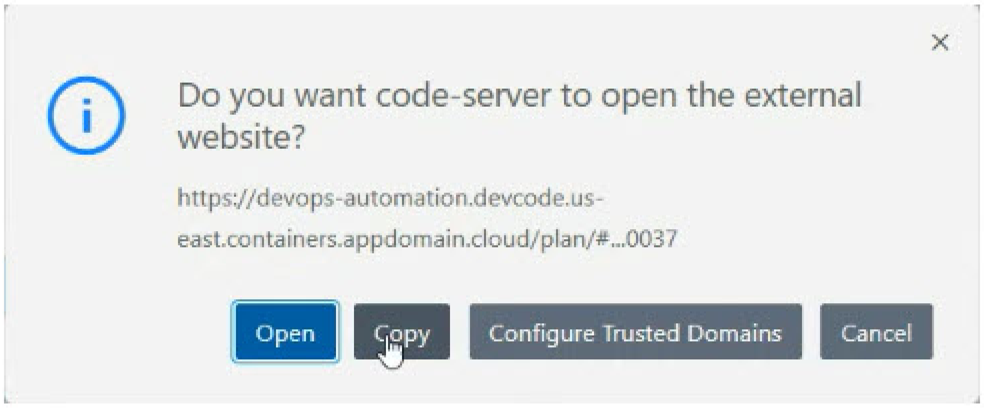
b. Fill in the required fields such as Application Project, Component, Type, Title, and Description.
c. Click the **Create** button to create your work item in Plan.


Creation of work item is successful

Source: DevOps Code

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.


Do you want code-server to open the external website?

https://devops-automation.devcode.us-east.containers.appdomain.cloud/plan/#...0037

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 DevOps 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.

Automatic Port Forwarding

DevOps Code (Code) automatically forwards ports from the dev container, so that you can access to applications that use TCP ports, from your local machine. For example, if you run a web application in Code, the port on which the web server listens will be forwarded so that you can access the web application from your web browser.

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 Management features

DevOps Code (Code) offers file management features such as drag-and-drop, browsing and downloading files, thereby simplifying your development workflow.

The following section outlines the key features provided by Code for file management:

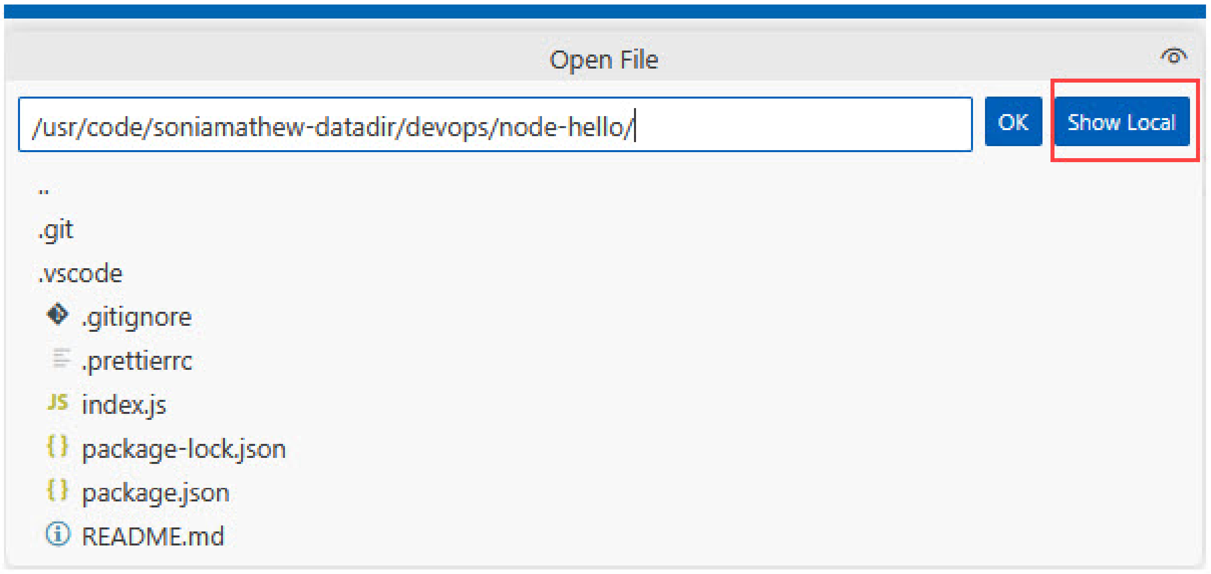
Drag and Drop

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

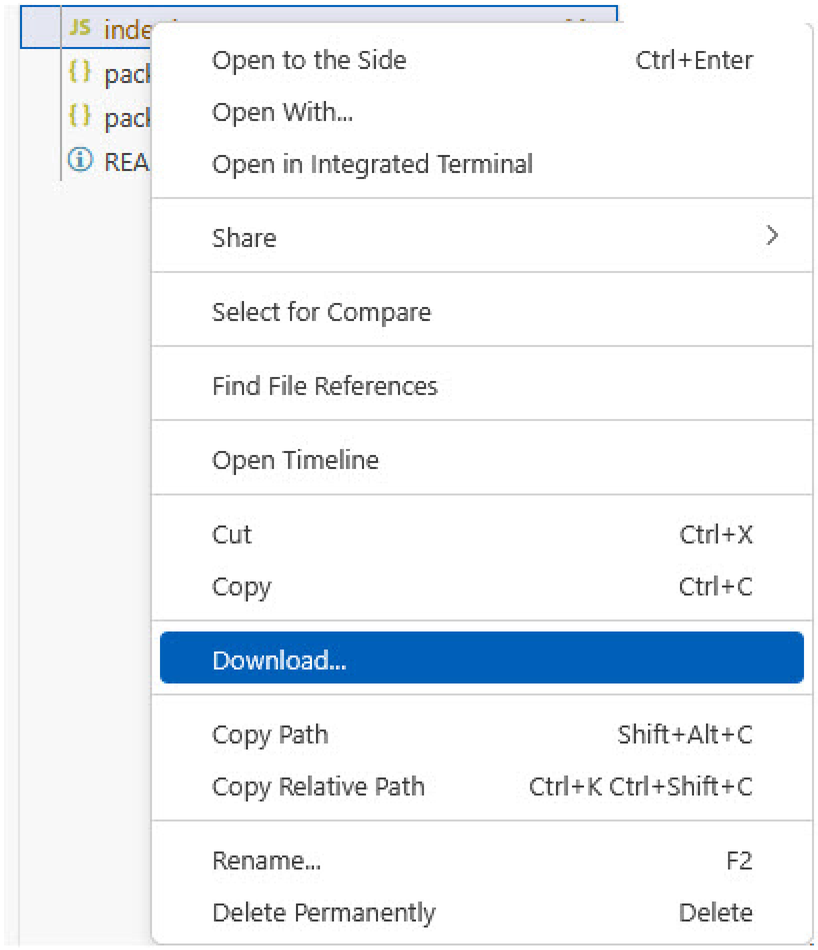
Code provides the **Show Local** button feature, which allows you to browse files from your local machine. This gives you the option to choose between using the file system of your local computer or that of the dev container.



```
/usr/code/soniamathew-datadir/devops/node-hello/
..
.git
.vscode
.gitignore
.prettierrc
index.js
package-lock.json
package.json
README.md
```

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.


inde
pack
pack
REA
Open to the Side Ctrl+Enter
Open With...
Open in Integrated Terminal
Share
Select for Compare
Find File References
Open Timeline
Cut Ctrl+X
Copy Ctrl+C
Download...
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Permanently Delete

Chapter 5. Troubleshooting

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.

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 Automation.

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

Trademarks

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

Rights

Index