

Deployment Guide for HCL CDP v12.1.8 Core Component on OpenShift



Contents

Chapter 1. Introduction.....	3
Chapter 2. Prerequisites.....	4
Chapter 3. List of CDP Core Components for OpenShift Cluster.....	5
Chapter 4. Installing Devtron.....	6
Chapter 5. Deploying Core Components.....	10
Deploying CDP Admin UI.....	10
Deploying CDP Admin UI Backend.....	13
Deploying CDP Core API.....	17
Deploying CDP Dash Backend.....	21
Deploying CDP Dash IFrame.....	24
Deploying CDP Web App.....	28
Deploying CDP Manager.....	31
Deploying CDP Pixel Listener.....	34
Deploying CDP DI API.....	37
Deploying CDP RTS.....	41
Deploying CDP RTS MD.....	45
Deploying CDP DMP SST.....	48
Deploying CDP DMP SST Maps.....	52
Deploying CDP DMP SST API.....	55
Deploying CDP DMP SST Zookeeper.....	58
Deploying CDP Live Events.....	60
Deploying CDP PostgreSQL API.....	63
Deploying CDP S3 Sink Connector.....	66
Deploying CDP Mongo Sink Connector.....	70
Deploying CDP KMS Service.....	74
Deploying CDP Mongo Connector Sink Job.....	77
Deploying CDP S3 Connector Sink Job.....	89

Chapter 1. Introduction

This section details the installation and configuration of the services and components required by HCL CDP core components deployment using devtron, on the Red Hat OpenShift platform.

Chapter 2. Prerequisites

Micro-services based HCL CDP core components, to be deployed on containerized platform using Devtron CI/CD tool (Red Hat OpenShift is the choice of the Container orchestration platform).

Micro-services based HCL CDP core components, to be deployed on containerized platform using Devtron CI/CD tool (Red Hat OpenShift is the choice of the Container orchestration platform).

Before proceeding with the installation of the required components, you have to ensure that Red Hat OpenShift is installed and properly configured. Refer to the Red Hat OpenShift documentation for installation guidelines.

Also, below tools are expected to present in the deployment to facilitate the installation:

- **OpenShift CLI:** Ensure the `oc` command is accessible and functional.
- **Kubernetes CLI:** Verify the availability of the `kubectl` command.
- **Helm CLI:** Ensure the `helm` command-line tool is installed.
- **Devtron Deployment Tool:** Confirm that Devtron is set up for managing deployments.

Chapter 3. List of CDP Core Components for OpenShift Cluster

List of HCL CDP Core modules components to be deployed and configured. CDP utilizes the functionalities from these modules to perform the end to end operations.

Application/Services	Deployment Type
CDP Admin UI	Helm chart
CDP Admin UI Backend	Helm chart
CDP Core API	Helm chart
CDP Dash Backend	Helm chart
CDP Dash Iframe	Helm chart
CDP DMP SST	Helm chart
CDP DMP SST Maps	Helm chart
CDP DMP SST API	Helm chart
CDP DMP SST Zookeeper	Helm chart
CDP Web App	Helm chart
CDP Live Events	Helm chart
CDP Manager	Helm chart
CDP Pixel Listener	Helm chart
CDP RTS	Helm chart
CDP RTS MS	Helm chart
CDP DI API	Helm chart
CDP Psql API	Helm chart
CDP S3 Sink Connector	Helm chart
CDP MongoDB Sink Connector	Helm chart
CDP KMS Service	Helm chart
DMP Producer	VM Based Deployment
NIFI	VM Based Deployment

Chapter 4. Installing Devtron

This section provides detailed instructions on how to deploy Devtron to your OpenShift cluster using Helm charts.

To install Devtron, follow these steps:

1. Add the Devtron repository to the helm chart sources.

```
helm repo add devtron https://helm.devtron.ai
helm repo update devtron
```

2. Install the Devtron on the OpenShift Cluster.

```
helm install devtron devtron/devtron-operator --create-namespace --namespace devtroncd --set
installer.arch=multi-arch --set installer.openshift=true
```



Note: Before installing the devtron, make sure to allow appropriate permissions in SCC.

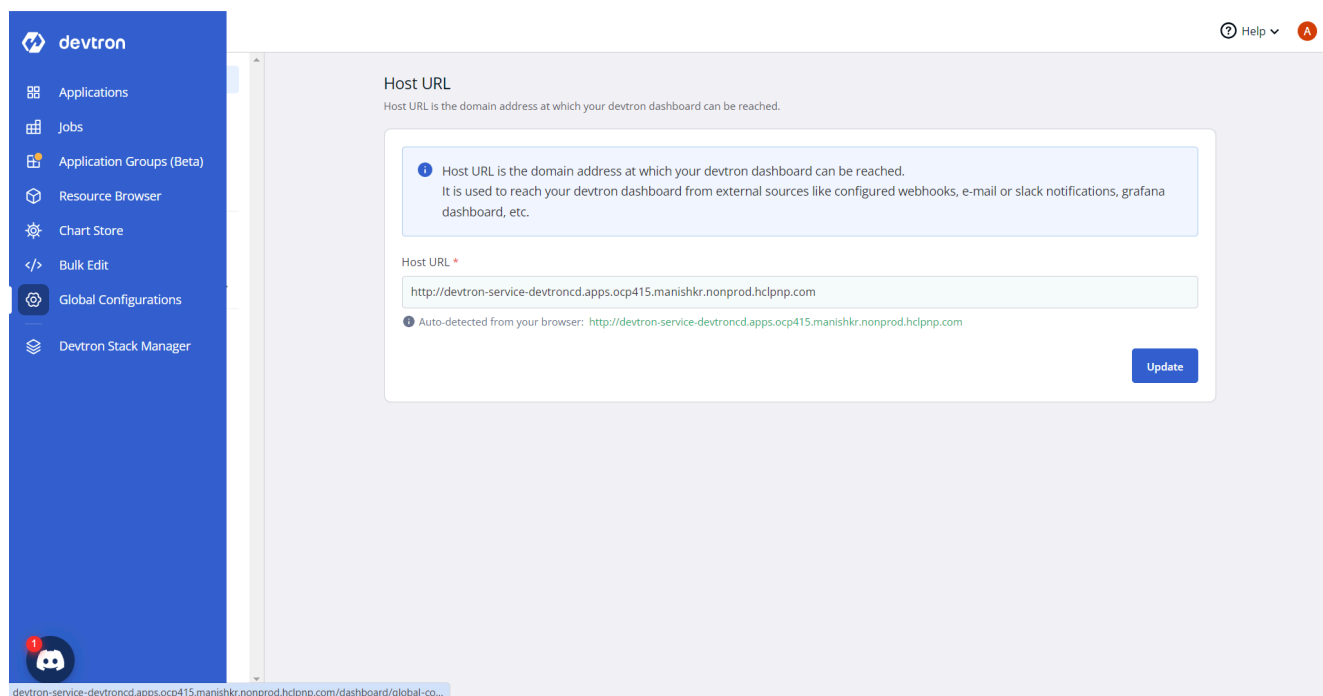
3. Get the Devtron login credentials. By default, the username will be admin.

```
kubectl -n devtroncd get secret devtron-secret -o jsonpath='{.data.ADMIN_PASSWORD}' | base64 -d
```

Configuring Devtron

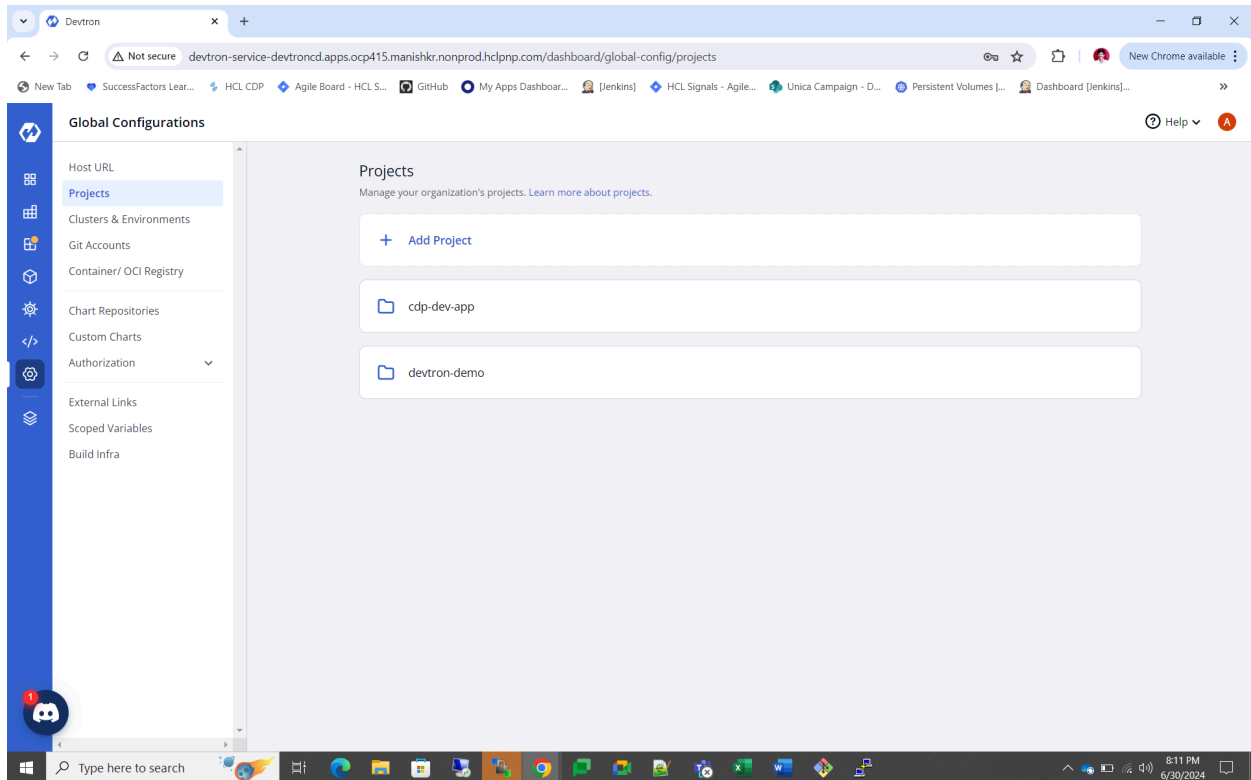
Prerequisites:

Before you start configuring the Devtron, verify the Global Configurations.

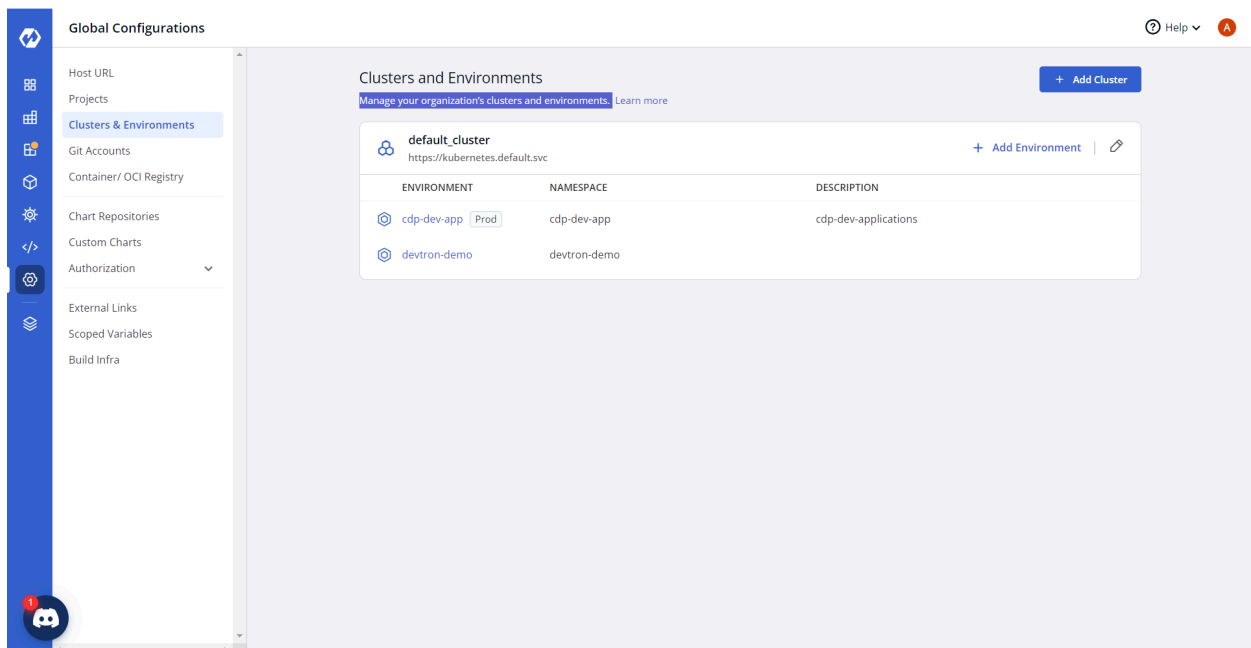


To configure the Devtron, follow the steps below:

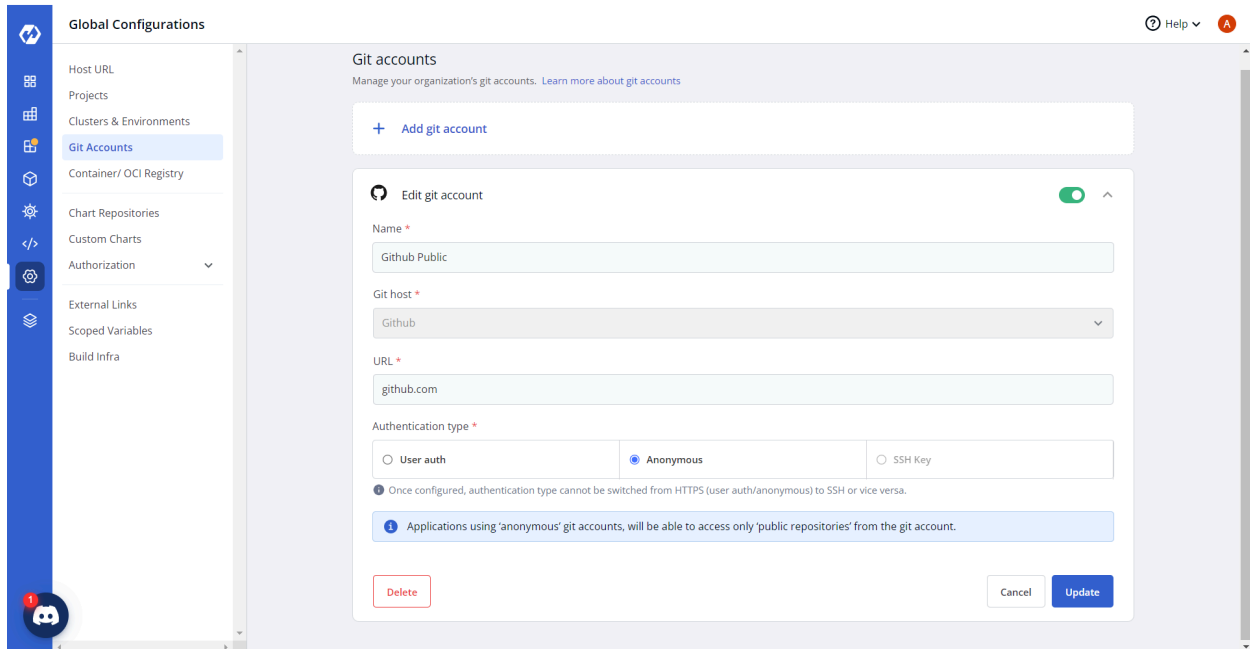
1. In the Devtron console, select **Global Configurations > Projects**. Click **Add Projects** and enter projects details.



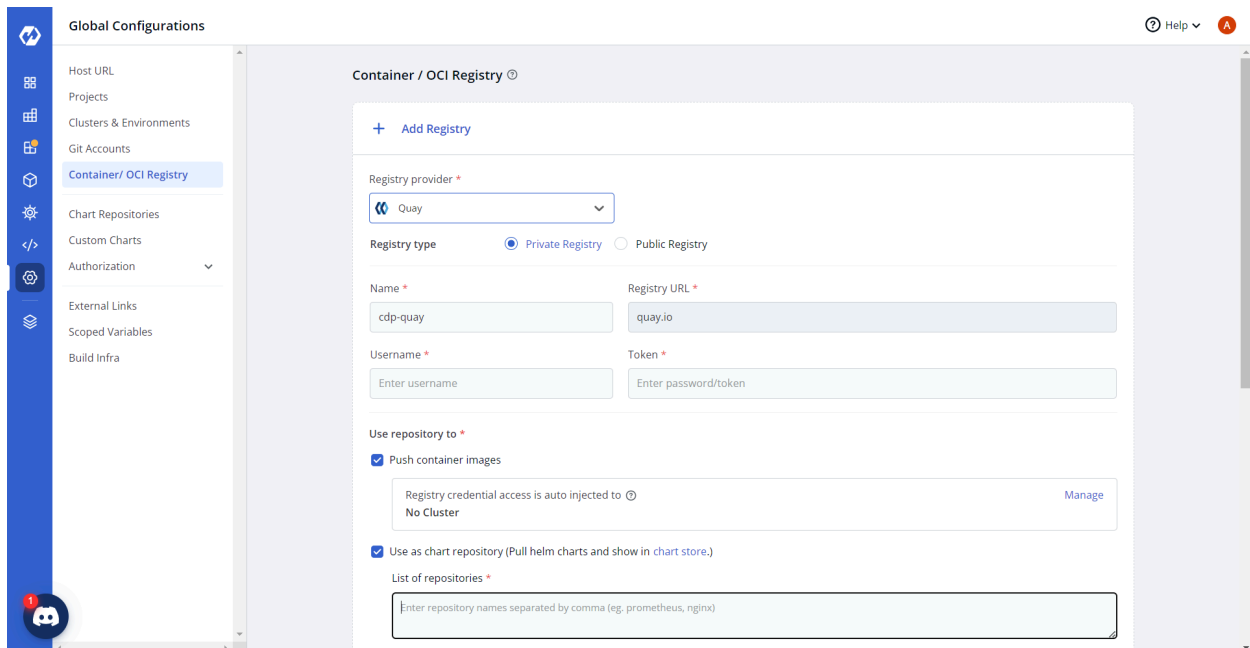
2. From the left pane, click **Clusters and Environment**, and manage your organizations clusters and environments details.



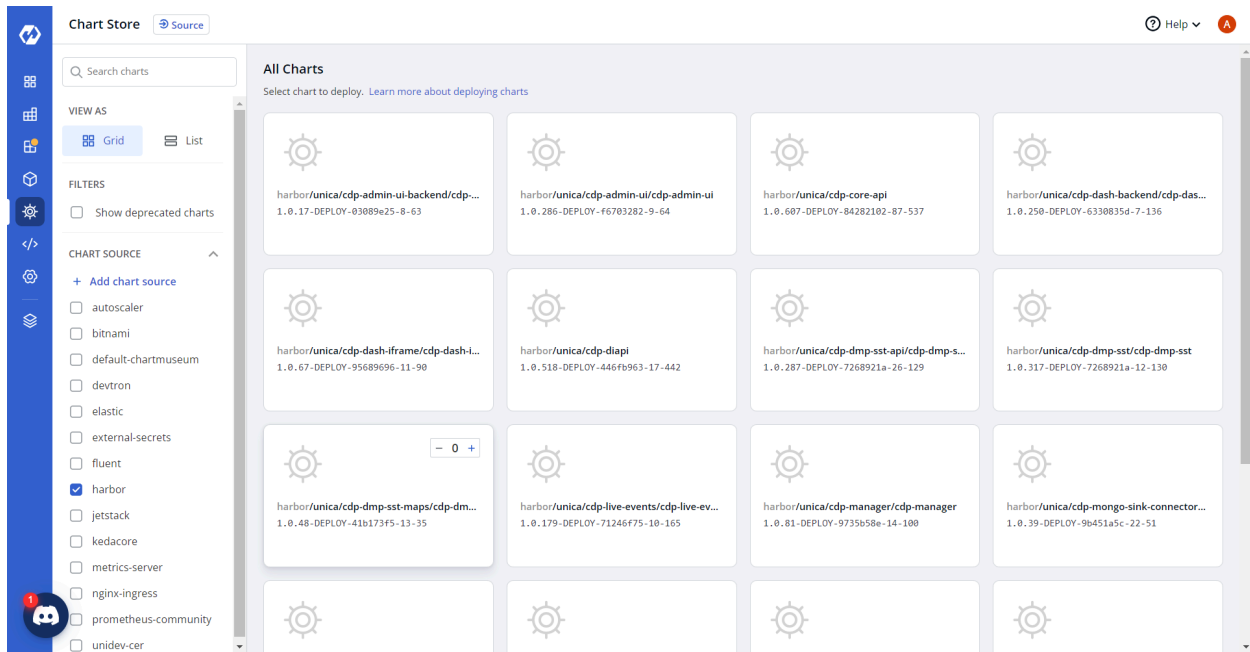
3. Click **Git Accounts**, and manage your organizations git accounts.



4. Click **Container/OCI Registry** > **Add Registry**, and update registry details and push helm charts and docker images to the configured registry.



5. In the registry configuration, update the list of repositories, and verify helm charts are synced to Devtron's chart store as shown below.



Chapter 5. Deploying Core Components

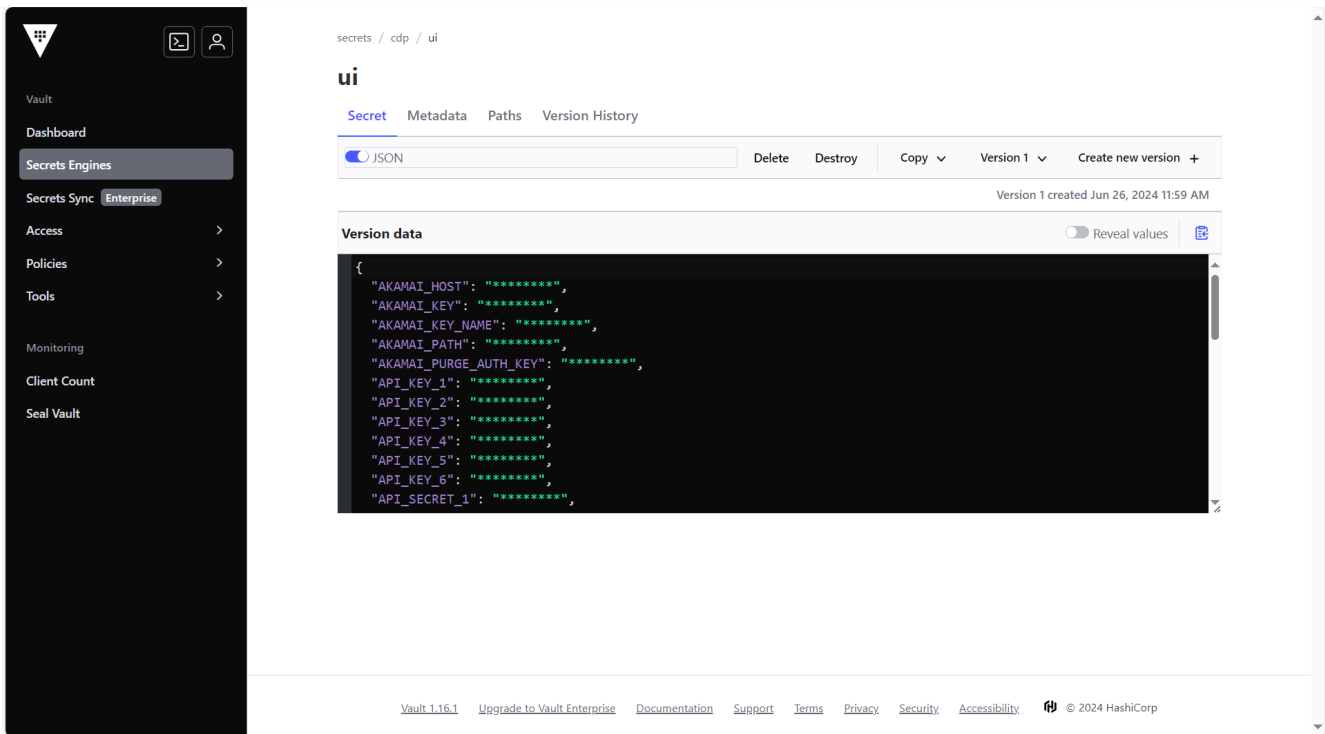
This section provides step-by-step instructions for deploying each service, component, and module required for HCL CDP Core Components. Follow the guidelines in each subsection to ensure the successful deployment and configuration of the respective elements.

Deploying CDP Admin UI

This section provides detailed instructions on how to deploy HCL CDP Admin UI using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP admin UI.



To create the UI secret in HashiCorp Vault, follow the steps below:

1. Create a UI secret sample key and value in the UI secret shown below, and update the actual values.

```
{
  "AKAMAI_HOST": "",
  "AKAMAI_KEY": "",
  "AKAMAI_KEY_NAME": "",
  "AKAMAI_PATH": "",
  "AKAMAI_PURGE_AUTH_KEY": "",
  "API_KEY_1": "",
  "API_KEY_2": "",
  "API_KEY_3": "",
  "API_KEY_4": "",
```

```

"API_KEY_5": "",
"API_KEY_6": "",
"API_SECRET_1": "",
"API_SECRET_2": "",
"API_SECRET_3": "",
"API_SECRET_4": "",
"API_SECRET_5": "",
"API_SECRET_6": "",
"AUTH_TOKEN": "",
"NEXT_PUBLIC_CLIENT_ID": "",
"NEXT_PUBLIC_CLIENT_SECRET": "",
"NEXT_PUBLIC_FROALA_SECRET": "",
"NEXT_PUBLIC_GOOGLE_CLIENT_ID": "",
"NEXT_PUBLIC_GOOGLE_CLIENT_SECRET": "",
"NEXT_PUBLIC_RECAPTCHA_KEY": "",
"REACT_APP_CLIENT_ID": "",
"REACT_APP_S3_ACCESS_KEY_ID": "",
"REACT_APP_S3_SECRET_KEY": "",
"auAuthKey": "",
"auEndpoint": "",
"host": "",
"muAuthKey": "",
"muEndpoint": "",
"password": "",
"sesEmail": "",
"sesPassword": "",
"smtpFrom": "",
"smtpHost": "",
"smtpPassword": "",
"smtpPort": "",
"smtpUser": "",
"usAuthKey": "",
"usEndpoint": "",
"user": ""
}

```

2. Create a Kubernetes Secret with Vault credentials to fetch the secrets from the vault.

```

apiVersion: v1
kind: Secret
metadata:
  name: vault-creds
  namespace: cdp-dev-app
type: Opaque
data:
  username: <vault username>
  password: <vault password>

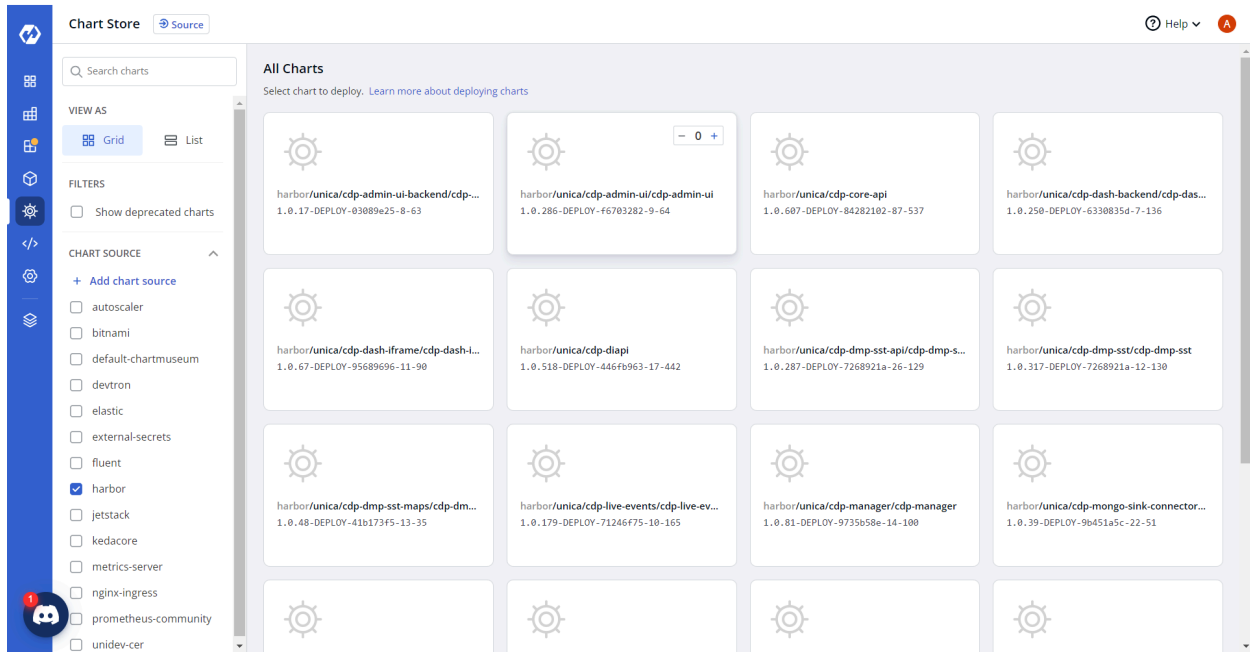
```

3. Create separate service account say cdp-dev-app-sa, with appropriate roles and permissions, and update ConfigMaps data with actual values.

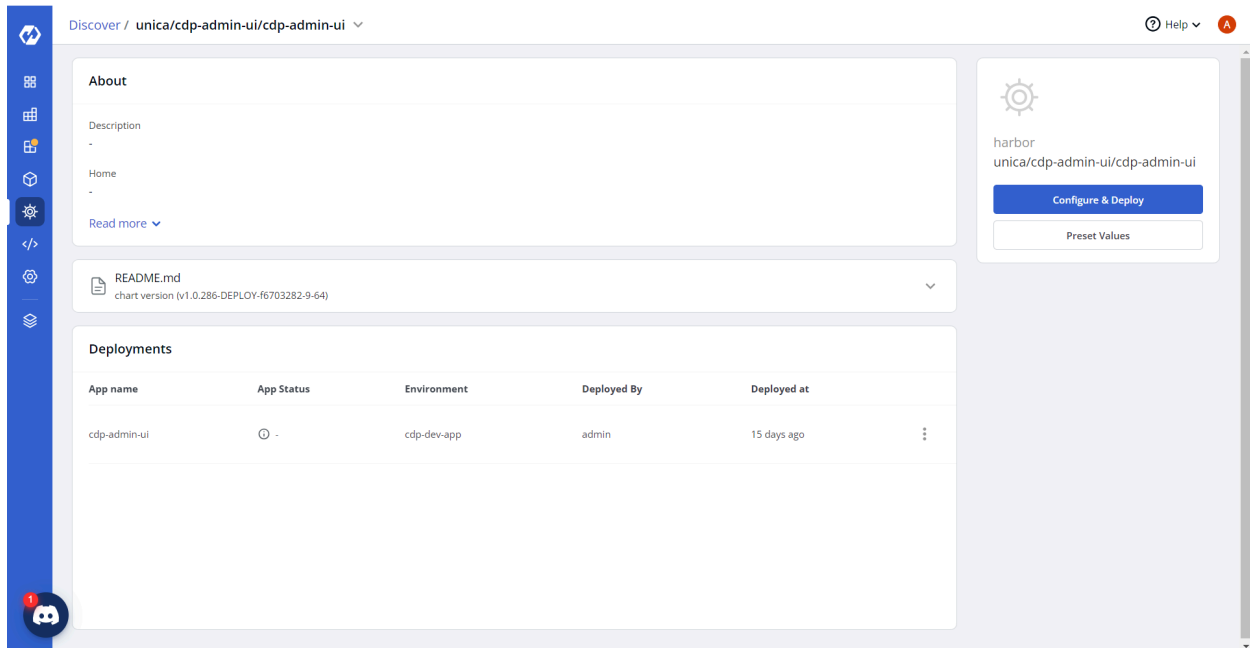
Deploying CDP Admin UI

To deploy the CDP admin UI, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-admin-ui chart to deploy.



2. Now, configure and deploy the admin UI charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```

REACT_APP_ADMIN_UI_BACKEND: <ADMIN_UI_BACKEND_URL>
REACT_APP_BYPASS_GOOGLE_OAUTH: "0"
REACT_APP_CAMPAIGN_ID: <VIZVRM6053>
REACT_APP_REST_API: '""'

```


Discover / unica/cdp-admin-ui/cdp-admin-ui

YAML Manifest output

App Name *
cdp-admin-ui

Project *
cdp-dev-app

Deploy to environment *
cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version
1.0.286-DEPLOY-6703282-9-64

Chart Values
Defa... (1.0.286-DEPLOY-6703282-9-64)

```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5         REACT_APP_ADMIN_UI_BACKEND: https://adminbackend.dev.hxcd.now.hclsoftware.cloud/api
6         REACT_APP_BYPASS_GOOGLE_OAUTH: "0"
7         REACT_APP_CAMPAIGN_ID: VIZVRM6053
8         REACT_APP_REST_API: ""
9     esoSecretData: {}
10  external: false
11  externalType: ""
12  filePermission: ""
13  mountPath: ""
14  name: admin-ui
15  roleARN: ""
16  subPath: false
17  type: environment
18 ConfigSecrets:
19   enabled: false
20   secrets: []
21 ContainerPort:
22   - envoyPort: 8799
23     idleTimeout: 1800s
24   name: app
25   port: 3000
26   servicePort: 80
27   supportStreaming: false
28   useHTTP2: false
29 EnvVariables: []
30 EnvVariablesFromConfigMapKeys: []
31 EnvVariablesFromFieldPath: []
32 EnvVariablesFromSecretKeys: []
33 GracePeriod: 30
34 livenessProbe:

```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

Helm Apps / cdp-admin-ui

Overview App Details Configure Deployment history

ENV cdp-dev-app

Application Status Healthy

Config apply status Deployed

Last updated 15 Days Ago

Chart used cdp-admin-ui

Version 1.0.286-... Notes.txt

K8s Resources Log Analyzer

All (10) Healthy (7)

Workloads

Deployment

Pod

cdp-admin-ui

ReplicaSet

Networking

Config & Storage

Pod (1)
1 healthy

Name	Ready	Restarts	Age
cdp-admin-ui-6798d7d4df-xx2g6	1/1	0	5d 10h

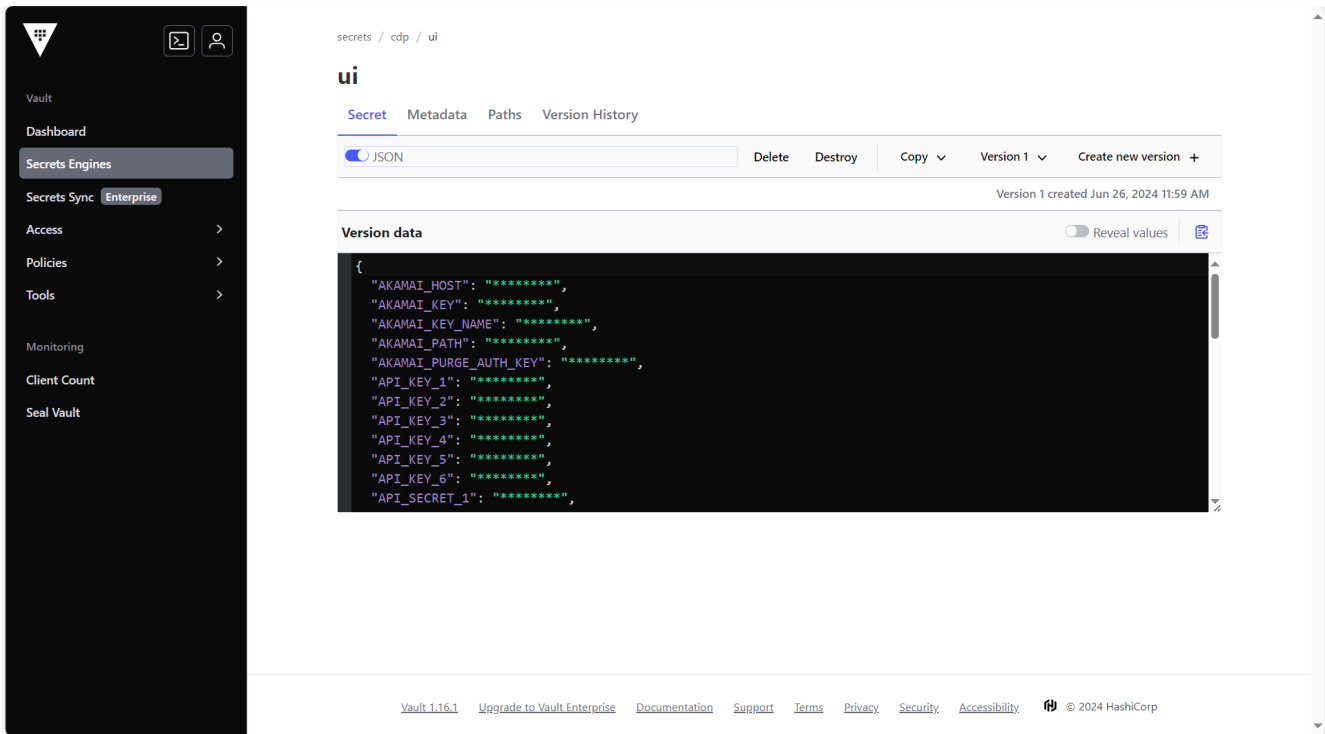
Log Analyzer

Deploying CDP Admin UI Backend

This section provides detailed instructions on how to deploy HCL CDP Admin UI Backend using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP admin UI.



To create the UI secret in HashiCorp Vault, follow the steps below:

1. Create a UI secret sample key and value in the UI secret shown below, and update the actual values.

```
{
  "AKAMAI_HOST": "",
  "AKAMAI_KEY": "",
  "AKAMAI_KEY_NAME": "",
  "AKAMAI_PATH": "",
  "AKAMAI_PURGE_AUTH_KEY": "",
  "API_KEY_1": "",
  "API_KEY_2": "",
  "API_KEY_3": "",
  "API_KEY_4": "",
  "API_KEY_5": "",
  "API_KEY_6": "",
  "API_SECRET_1": "",
  "API_SECRET_2": "",
  "API_SECRET_3": "",
  "API_SECRET_4": "",
  "API_SECRET_5": "",
  "API_SECRET_6": "",
  "AUTH_TOKEN": "",
  "NEXT_PUBLIC_CLIENT_ID": "",
  "NEXT_PUBLIC_CLIENT_SECRET": "",
  "NEXT_PUBLIC_FROALA_SECRET": "",
  "NEXT_PUBLIC_GOOGLE_CLIENT_ID": "",
  "NEXT_PUBLIC_GOOGLE_CLIENT_SECRET": "",
}
```

```

"NEXT_PUBLIC_RECAPTCHA_KEY": "",
"REACT_APP_CLIENT_ID": "",
"REACT_APP_S3_ACCESS_KEY_ID": "",
"REACT_APP_S3_SECRET_KEY": "",
"auAuthKey": "",
"auEndpoint": "",
"host": "",
"muAuthKey": "",
"muEndpoint": "",
"password": "",
"sesEmail": "",
"sesPassword": "",
"smtpFrom": "",
"smtpHost": "",
"smtpPassword": "",
"smtpPort": "",
"smtpUser": "",
"usAuthKey": "",
"usEndpoint": "",
"user": ""
}

```

2. Create a Kubernetes Secret with Vault credentials to fetch the secrets from the vault.

```

apiVersion: v1
kind: Secret
metadata:
  name: vault-creds
  namespace: cdp-dev-app
type: Opaque
data:
  username: <vault username>
  password: <vault password>

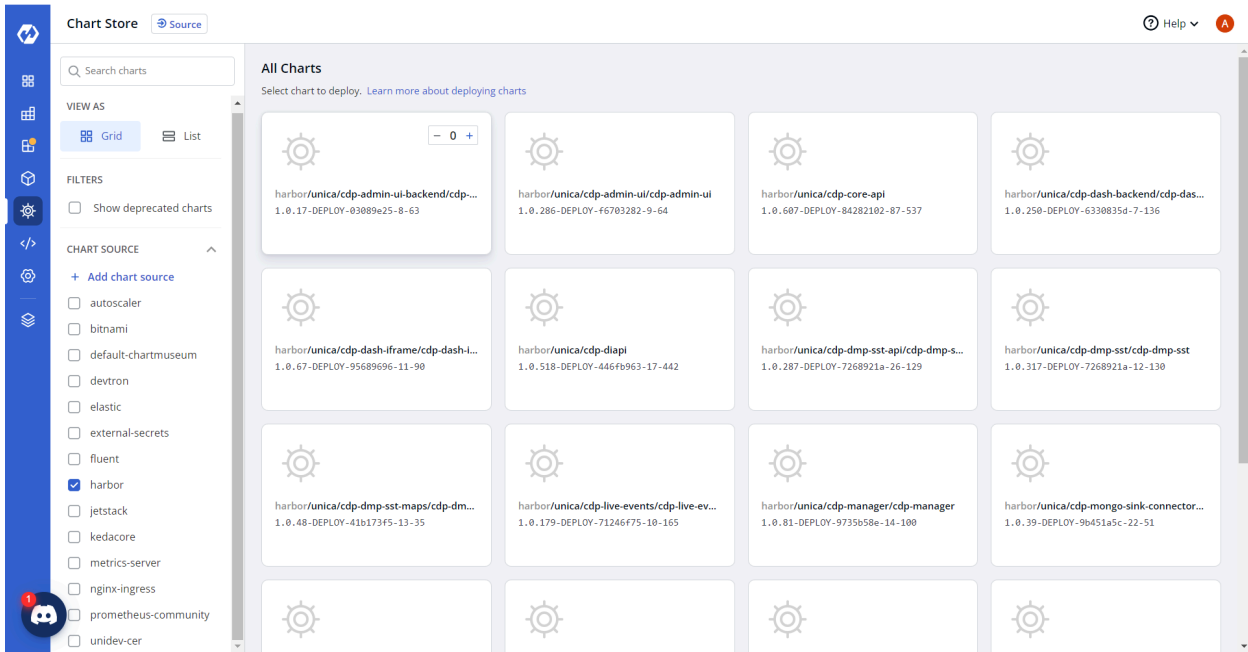
```

3. Create separate service account say cdp-dev-app-sa, with appropriate roles and permissions, and update ConfigMaps data with actual values.

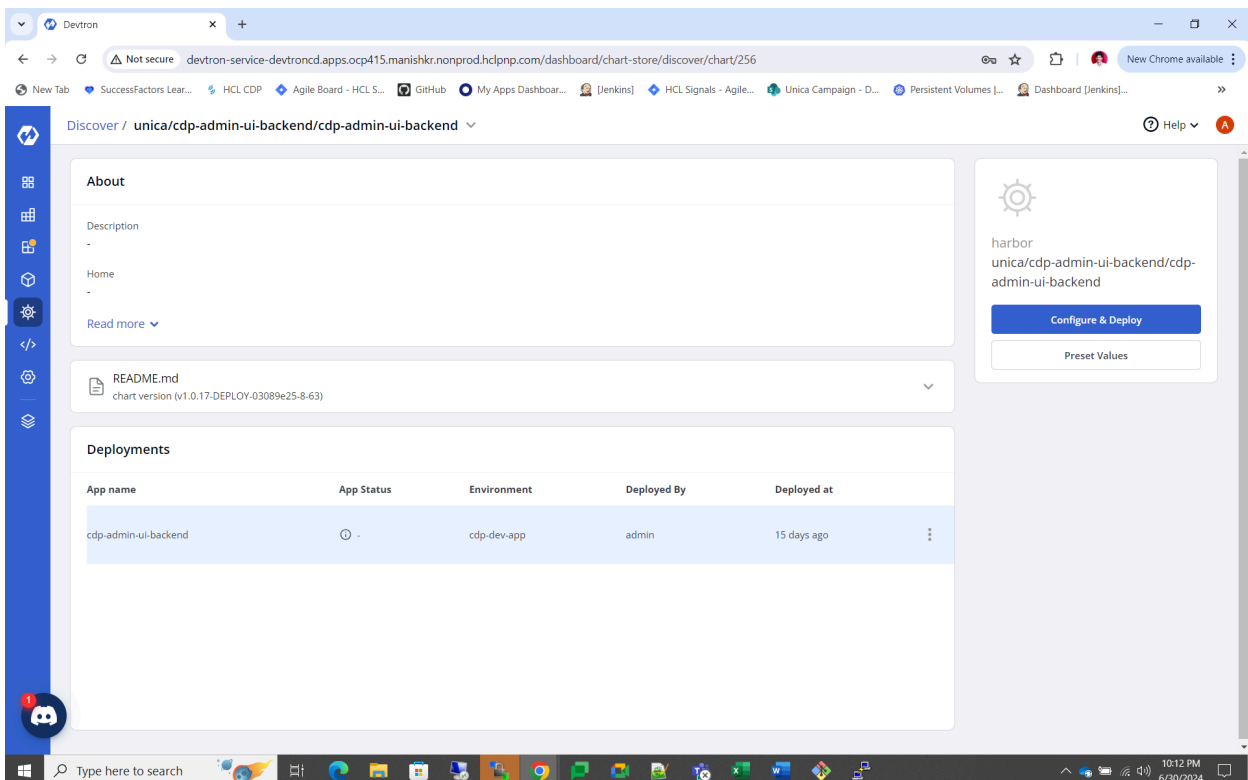
Deploying CDP Admin UI Backend

To deploy the CDP admin UI backend, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-admin-ui-backend chart to deploy.



2. Now, configure and deploy the admin UI charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```

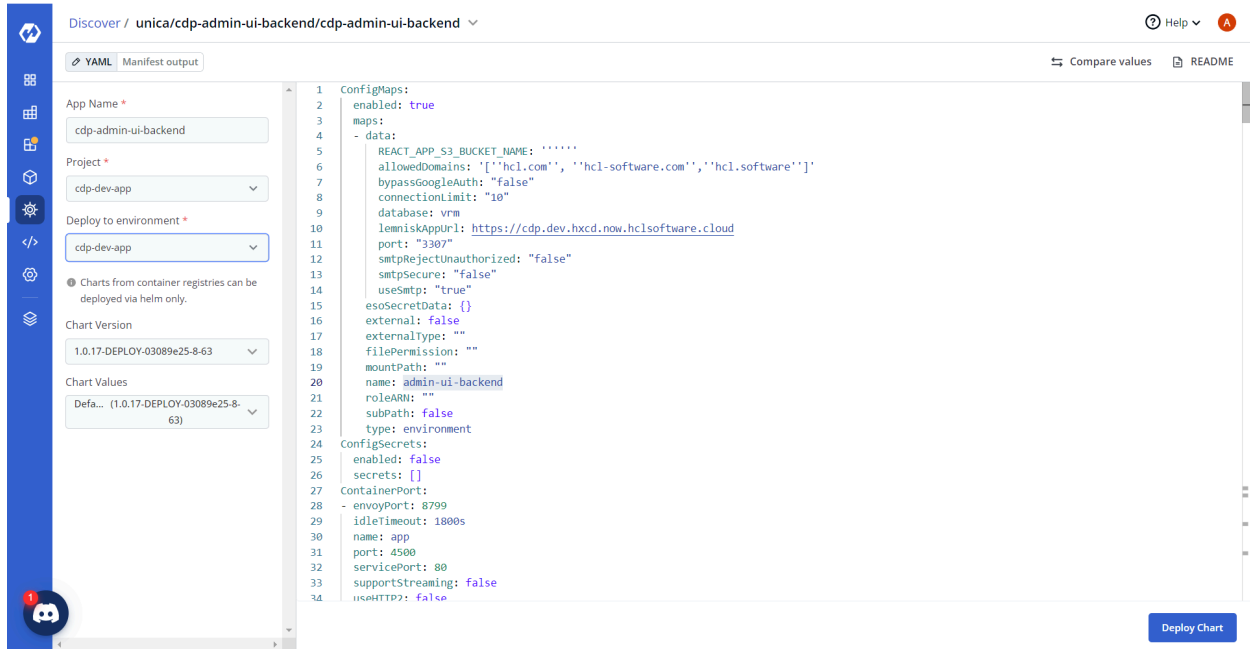
REACT_APP_S3_BUCKET_NAME: ''
allowedDomains: ['hcl.com', 'hcl-software.com', 'hcl.software', 'hclpnp.com']
bypassGoogleAuth: "false"
connectionLimit: "10"

```

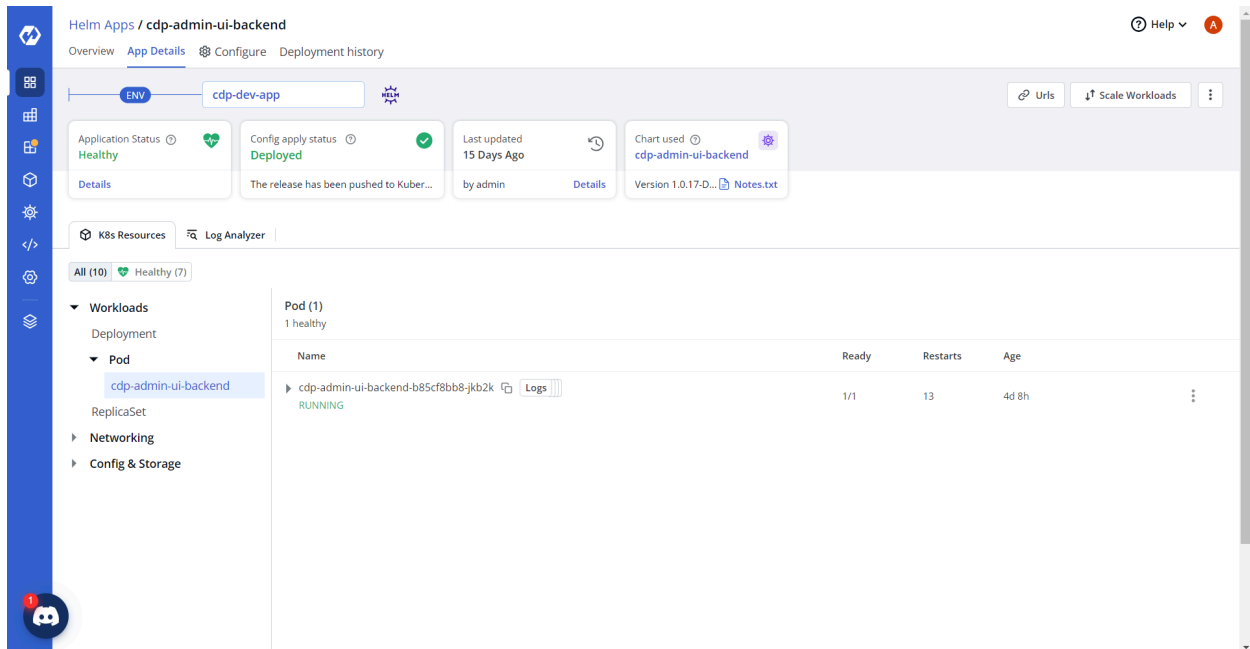
```

database:< MySQL_DB_Name>
lemniskAppUrl: <WebApp_URL>
port: "<3306>"
smtpRejectUnauthorized: "false"
smtpSecure: "false"
useSmtp: "true"

```



4. On successful deployment, validate the deployment as shown below.

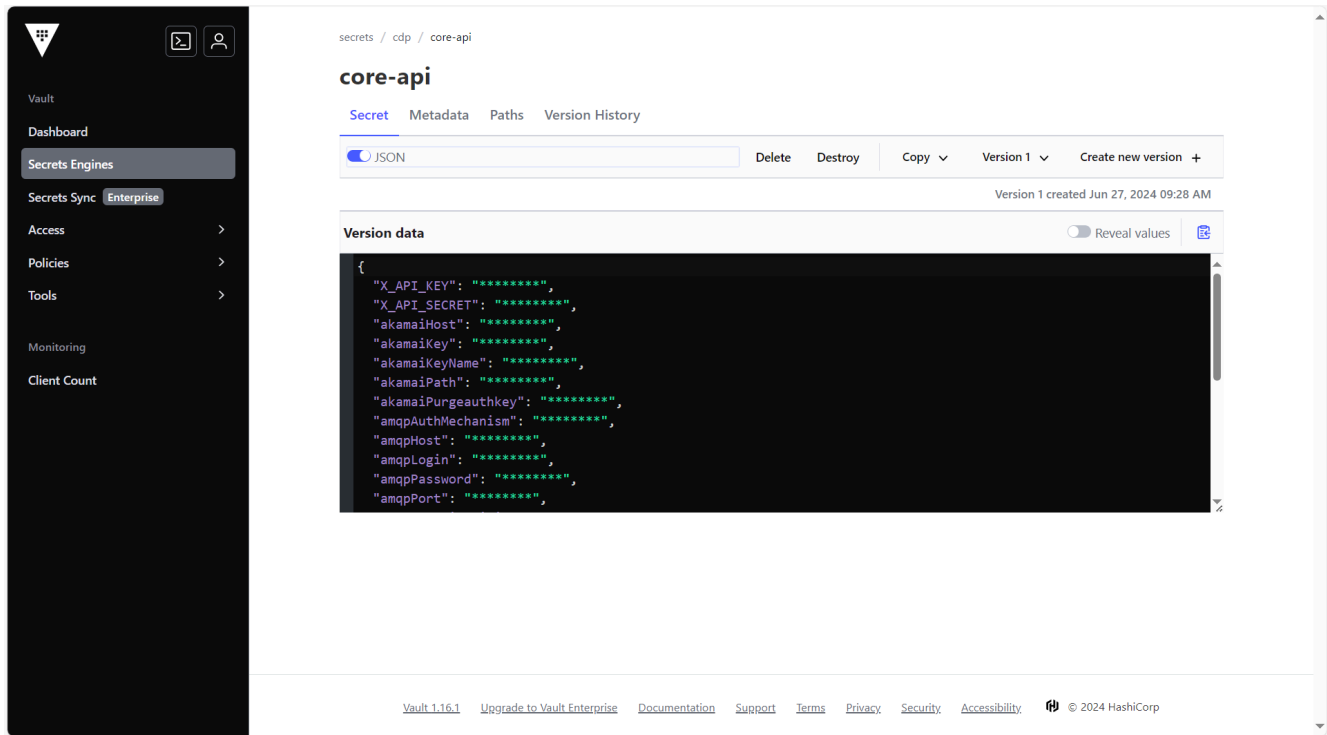


Deploying CDP Core API

This section provides detailed instructions on how to deploy HCL CDP Core API using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP core api.



To create the UI secret in HashiCorp Vault, follow the steps below:

1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "X_API_KEY": "\\\"",
  "X_API_SECRET": "\\\"",
  "akamaiHost": "\\\"",
  "akamaiKey": "\\\"",
  "akamaiKeyName": "\\\"",
  "akamaiPath": "\\\"",
  "akamaiPurgeauthkey": "\\\"",
  "amqpAuthMechanism": "\\\"",
  "amqpHost": "\\\"",
  "amqpLogin": "\\\"",
  "amqpPassword": "\\\"",
  "amqpPort": "\\\"",
  "dbConnectionLimit": "\\\"80\\\"",
  "dbDatabase": "\\\"<dbName>\\\"",
  "dbHost": "\\\"<dbHost>\\\"",
  "dbPassword": "\\\"<dbPassword>\\\"",
  "dbPort": "\\\"<dbPort>\\\"",
  "dbUser": "\\\"<dbUser>\\\"",
  "lemnisk_mailId": "\\\"",
  "lemnisk_mailPassword": "\\\"",
  "mailId": "\\\"",
  "mailPassword": "\\\""
```

```

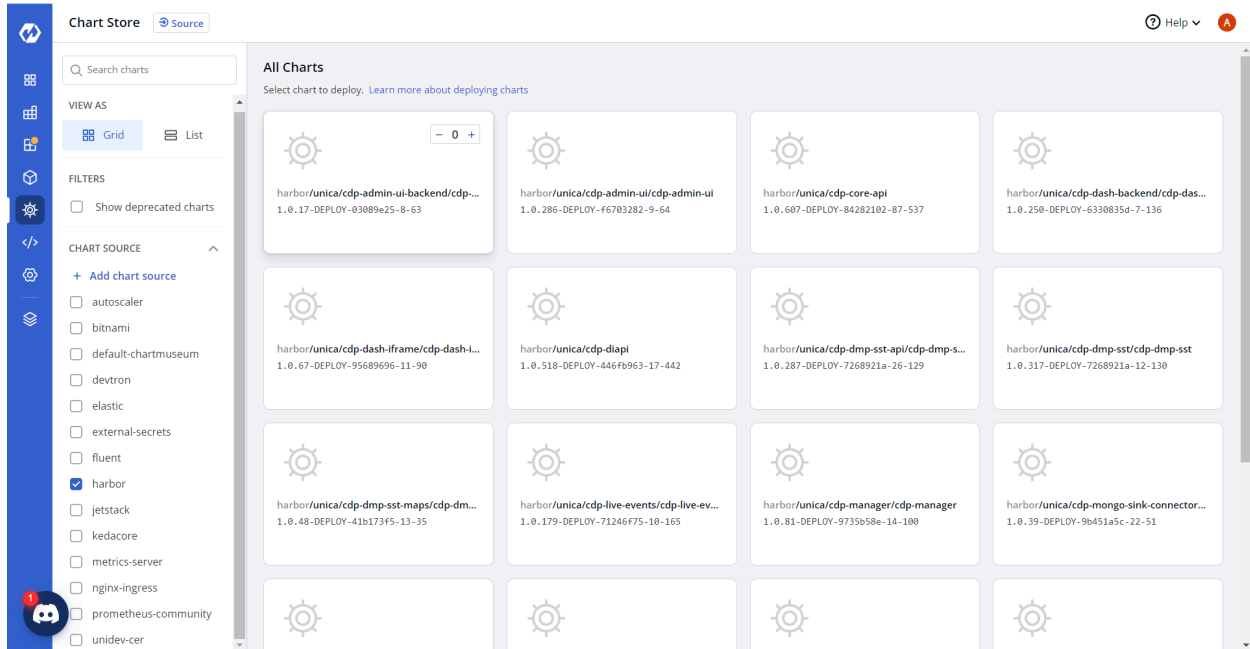
"stagingS3AccessKey": "\\\"",
"stagingSecretAccessKey": "\\\"",
"userIdentifierAuthKey": "\\<userIdentifierAuthKey>\\\""
}

```

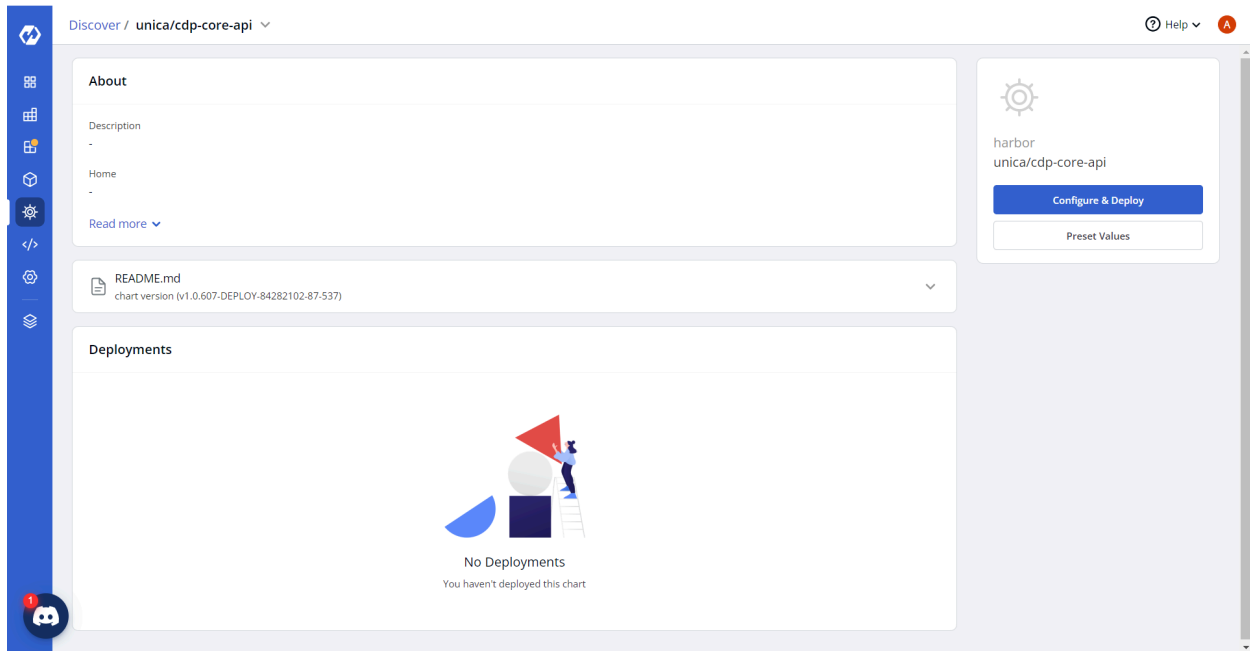
Deploying CDP CORE API

To deploy the CDP core API, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-core-api chart to deploy.



2. Now, configure and deploy the cdp core api charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
CampaignRegionList: '{aps1:['6525']}'
DEFAULT_SRC:
FONT_SRC:
IMG_SRC:
SCRIPT_SRC:
STYLE_SRC:
DOMAIN:
cdpDashboardService:
druidURL:
kms_decrypt:
kms_encrypt:
kms_region:
muUserIdentifierDomain:
psql_endpoint:
psql_host:
rts_endpoint:
s3Bucket:
usUserIdentifierDomain:
segmentUserListuploadsBucket:
uploadsBucket
```


Discover / unica/cdp-core-api

YAML Manifest output

App Name *
cdp-core-api

Project *
cdp-dev-app

Deploy to environment *
cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version
1.0.607-DEPLOY-84282102-87-537

Chart Values
Def... (1.0.607-DEPLOY-84282102-87-537)

```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5         API: .....
6         CampaignRegionList: '{aps1:['6525']}'
7         DEFAULT_SRC: 'http://*.apps.ocp415.manishkr.nonprod.hclpnp.com:* https://*.apps.ocp415.manishkr.nonprod.hclpnp.com:*
8           http://*.dev.hxcd.now.hclsoftware.cloud:* https://*.dev.hxcd.now.hclsoftware.cloud:*
9           http://*.dev.hxcb.now.hclsoftware.cloud:* https://*.dev.hxcb.now.hclsoftware.cloud:*
10          https://www.google-analytics.com http://*.hclsoftware.cloud:* https://*.hclsoftware.cloud:*'
11         DSN: .....
12         FONT_SRC: 'http://*.apps.ocp415.manishkr.nonprod.hclpnp.com:* https://*.apps.ocp415.manishkr.nonprod.hclpnp.com:*
13           http://*.dev.hxcd.now.hclsoftware.cloud:* https://*.dev.hxcd.now.hclsoftware.cloud:*
14           http://*.dev.hxcb.now.hclsoftware.cloud:* https://*.dev.hxcb.now.hclsoftware.cloud:*
15           https://www.google-analytics.com http://*.hclsoftware.cloud:* https://*.hclsoftware.cloud:*
16           fonts.gstatic.com data: ;'
17         IMG_SRC: 'http://*.apps.ocp415.manishkr.no Follow link (ctrl + click) :tps://*.apps.ocp415.manishkr.nonprod.hclpnp.com:*
18           http://*.dev.hxcd.now.hclsoftware.cloud:* https://*.dev.hxcd.now.hclsoftware.cloud:*
19           http://*.dev.hxcb.now.hclsoftware.cloud:* https://*.dev.hxcb.now.hclsoftware.cloud:*
20           https://www.google-analytics.com http://*.hclsoftware.cloud:* https://*.hclsoftware.cloud:*
21         data: blob;'
22         SCRIPT_SRC: 'http://*.apps.ocp415.manishkr.nonprod.hclpnp.com:* https://*.apps.ocp415.manishkr.nonprod.hclpnp.com:*
23           http://*.dev.hxcd.now.hclsoftware.cloud:* https://*.dev.hxcd.now.hclsoftware.cloud:*
24           http://*.dev.hxcb.now.hclsoftware.cloud:* https://*.dev.hxcb.now.hclsoftware.cloud:*
25           https://www.google-analytics.com http://*.hclsoftware.cloud:* https://*.hclsoftware.cloud:*
26           "unsafe-inline" "unsafe-eval"'
27         STYLE_SRC: 'http://*.apps.ocp415.manishkr.nonprod.hclpnp.com:* https://*.apps.ocp415.manishkr.nonprod.hclpnp.com:*
28           http://*.dev.hxcd.now.hclsoftware.cloud:* https://*.dev.hxcd.now.hclsoftware.cloud:*
29           http://*.dev.hxcb.now.hclsoftware.cloud:* https://*.dev.hxcb.now.hclsoftware.cloud:*
30           https://www.google-analytics.com http://*.hclsoftware.cloud:* https://*.hclsoftware.cloud:*
31           "unsafe-inline"
32         TTPActivationId: .....
33         accessTokenLifetime: '86400'
34         authDomain: .....

```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

Devtron

Not secure devtron-service-devtroncd.apps.ocp415.manishkr.nonprod.hclpnp.com/dashboard/app/dc/deployments/5/env/2/details/k8s-resources/pod/group/cdp-cor...

Helm Apps / cdp-core-api

Overview App Details Configure Deployment history

ENV cdp-dev-app

Application Status Healthy

Config apply status Deployed

Last updated 2 Days Ago

Chart used hbr-cdp-core-api

Details The release has been pushed to Kuber... by admin Details Version 1.0.621-... Notes.txt

K8s Resources Log Analyzer

All (10) Healthy (7)

Workloads

Deployment

Pod

cdp-core-api-hbr-cdp-core-api

ReplicaSet

Networking

Config & Storage

Pod (1)
1 healthy

Name	Ready	Restarts	Age
cdp-core-api-hbr-cdp-core-api-7874cbcb5b-lfhlf	1/1	0	2d

Log Analyzer

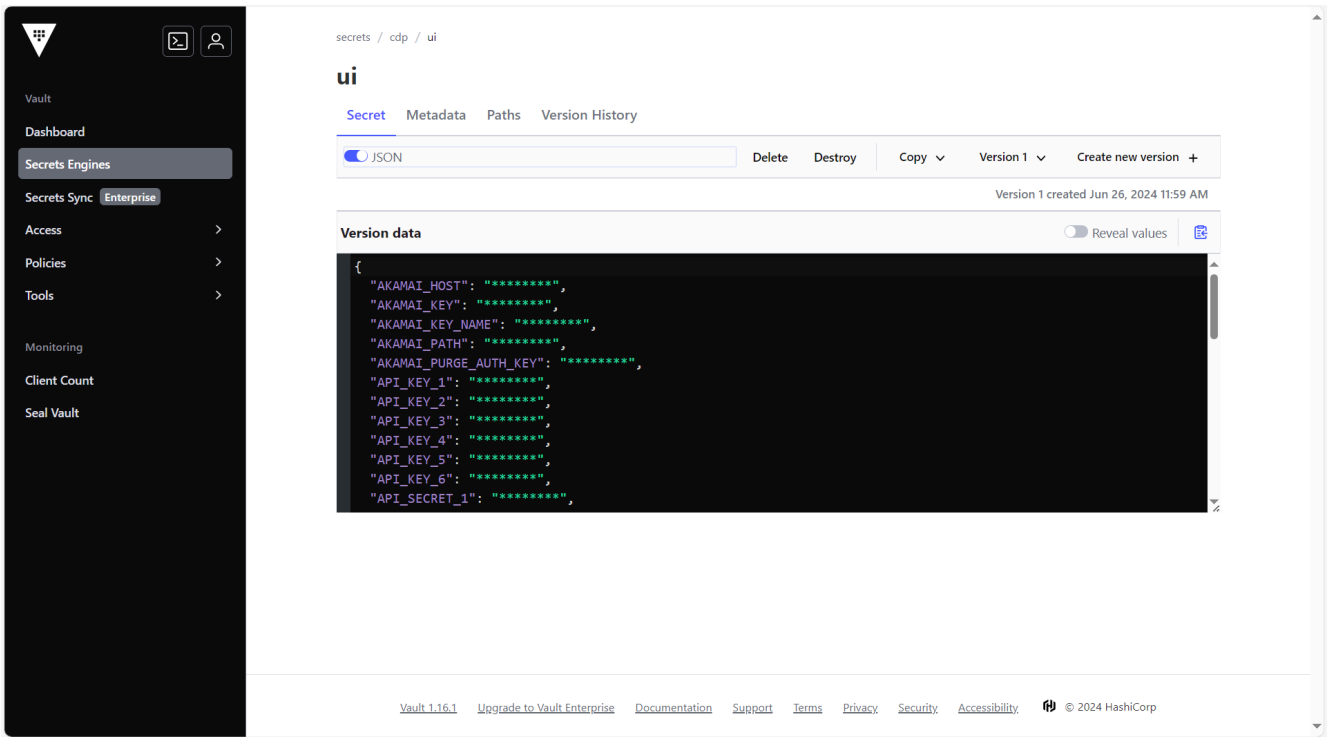
10:26 PM 6/30/2024

Deploying CDP Dash Backend

This section provides detailed instructions on how to deploy HCL CDP Dash Backend using the Devtron in the OpenShift.

Prerequisites:

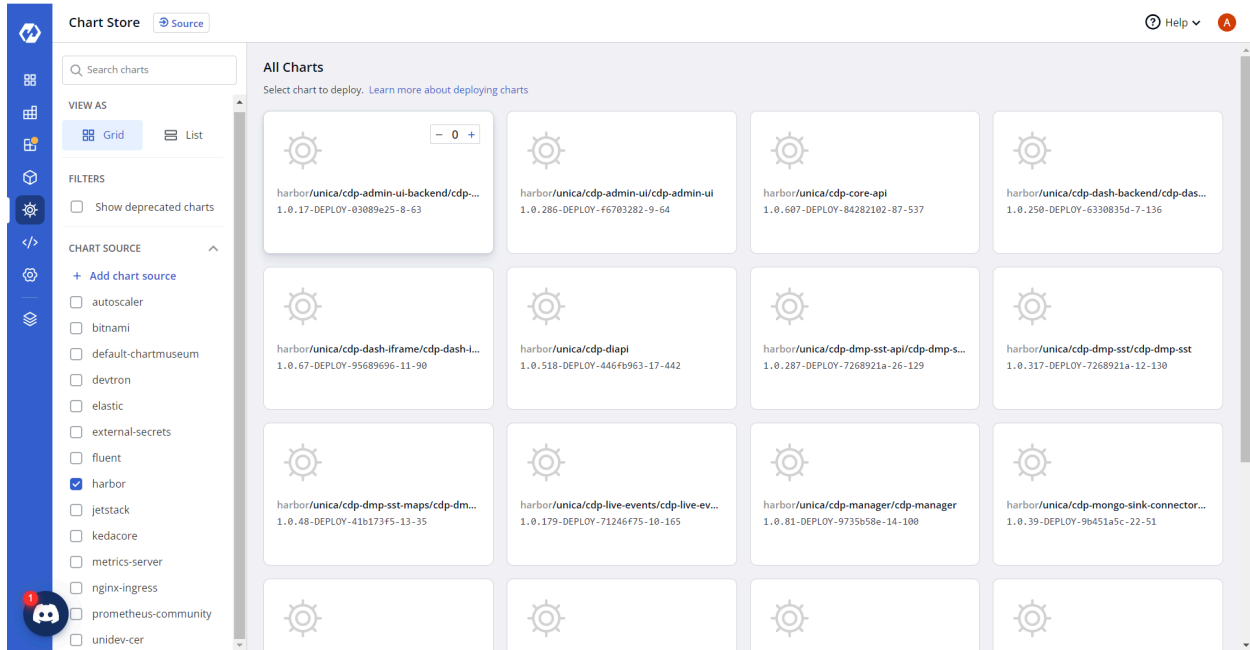
Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Dash Backend.



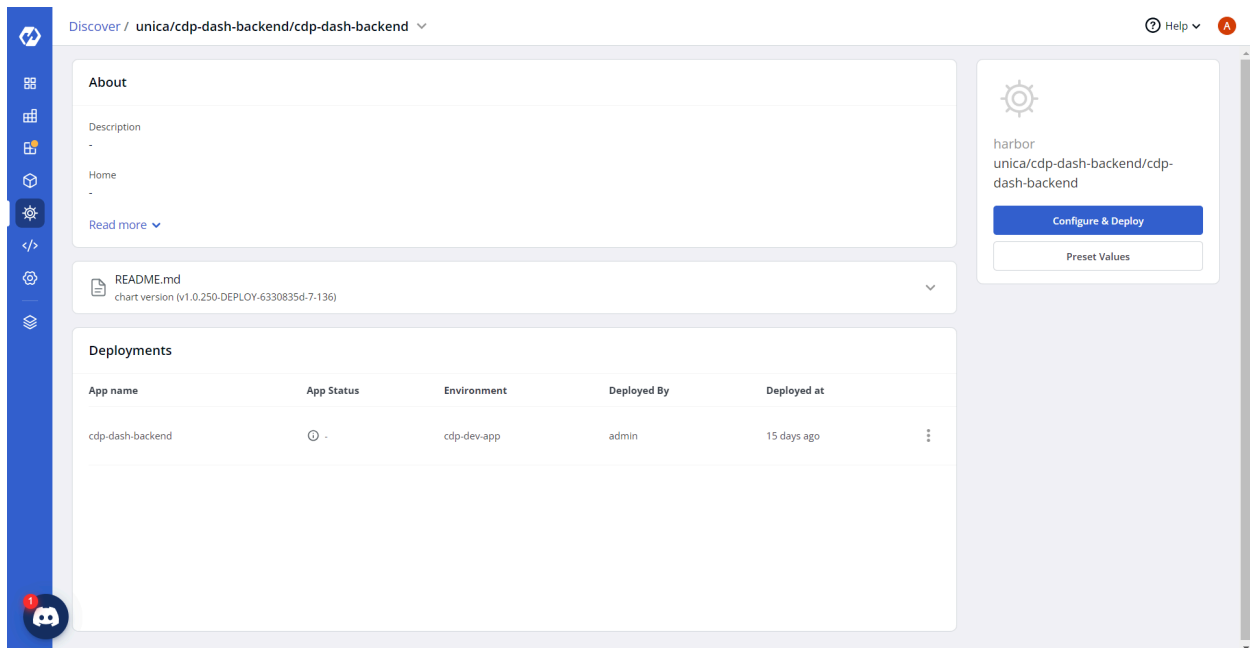
Deploying CDP Dash Backend

To deploy the CDP Dash Backend, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-dash-backend chart to deploy.



2. Now, configure and deploy the CDP Dash Backend charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
CORE_API:<http://core_api_url/-/v1/verifyUserAuthentication?access_token=>
CORS: '[cors_urls]'
DEMO_CAMPAIGN_ID: '[6525]'
DRUID_HOST: http://<druid_host:port>/druid/v2/sql
LOGGING_MODE: <live>
```

PORT: "<port>"
REGION: <region>

The screenshot shows the OpenShift console interface for the 'cdp-dash-backend' Helm chart. The left sidebar contains navigation icons. The main panel displays the 'YAML' view of the chart configuration. The configuration includes fields for App Name, Project, Deploy to environment, Chart Version, and Chart Values. The 'Manifest output' tab is active, showing the rendered Kubernetes manifest. The manifest includes ConfigMaps, ConfigSecrets, and a Deployment. The Deployment configuration specifies the container port, service port, and various environment variables.

```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5       CORE_API: https://coreapi.dev.hxcd.now.hclsoftware.cloud/-/v1/verifyUserAuthentication?access_token=
6       CORS: '["https://cdp.dev.hxcd.now.hclsoftware.cloud", "https://cdp.dev.hxcd.now.hclsoftware.cloud/app", "https://dash-iframe.dev.hxcd.now.hclsoftware.cloud"]'
7       DEMO_CAMPAIGN_ID: '[6525]'
8       DRUID_HOST: http://10.214.101.158:8888/druid/v2/sql
9       LOGGING_MODE: live
10      PORT: "4398"
11      REGION: ap-south-1
12      esoSecretData: {}
13      external: false
14      externalType: ""
15      filePermission: ""
16      mountPath: ""
17      name: cdp-dash-backend
18      roleARN: ""
19      subPath: false
20      type: environment
21 ConfigSecrets:
22   enabled: false
23   secrets: []
24 ContainerPort:
25   - envoyPort: 8799
26     idletimeout: 1800s
27   name: app
28   port: 4398
29   servicePort: 80
30   supportStreaming: false
31   useHTTP2: false
32 EnvVariables: []
33 EnvVariablesFromConfigMapKeys: []
34 EnvVariablesFromFieldPath: []
  
```

At the bottom right, there is a 'Deploy Chart' button.

4. On successful deployment, validate the deployment as shown below.

The screenshot shows the OpenShift console interface for the 'cdp-dash-backend' Helm chart. The left sidebar contains navigation icons. The main panel displays the 'Overview' tab of the 'cdp-dash-backend' Helm App. The overview shows the application status as 'Healthy', the configuration status as 'Deployed', and the last updated time as '15 Days Ago'. Below this, there is a table showing the deployment status of the pods. The table has columns for Name, Ready, Restarts, and Age. The pod 'cdp-dash-backend-d6b4c8ddd-4ftmv' is shown in a 'RUNNING' state with 1/1 replicas ready, 0 restarts, and an age of 15d 6h.

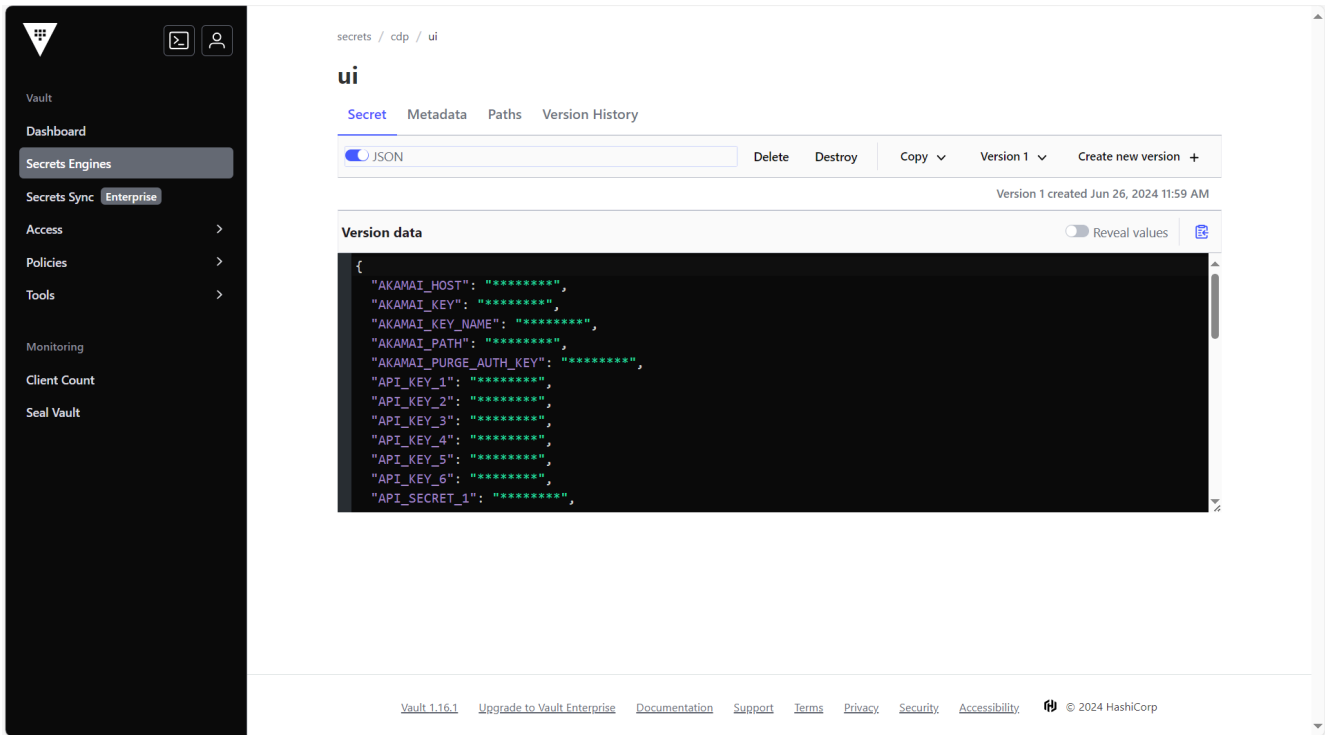
Name	Ready	Restarts	Age
cdp-dash-backend-d6b4c8ddd-4ftmv	1/1	0	15d 6h

Deploying CDP Dash IFrame

This section provides detailed instructions on how to deploy HCL CDP Dash IFrame using the Devtron in the OpenShift.

Prerequisites:

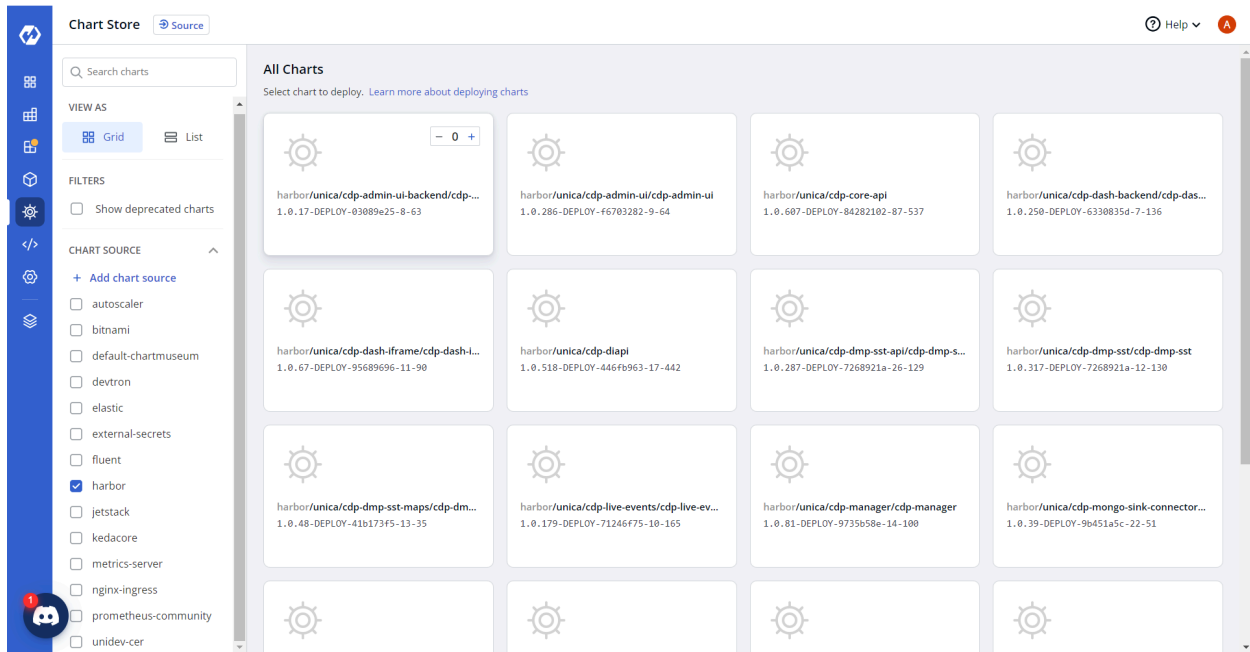
Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Dash IFrame.



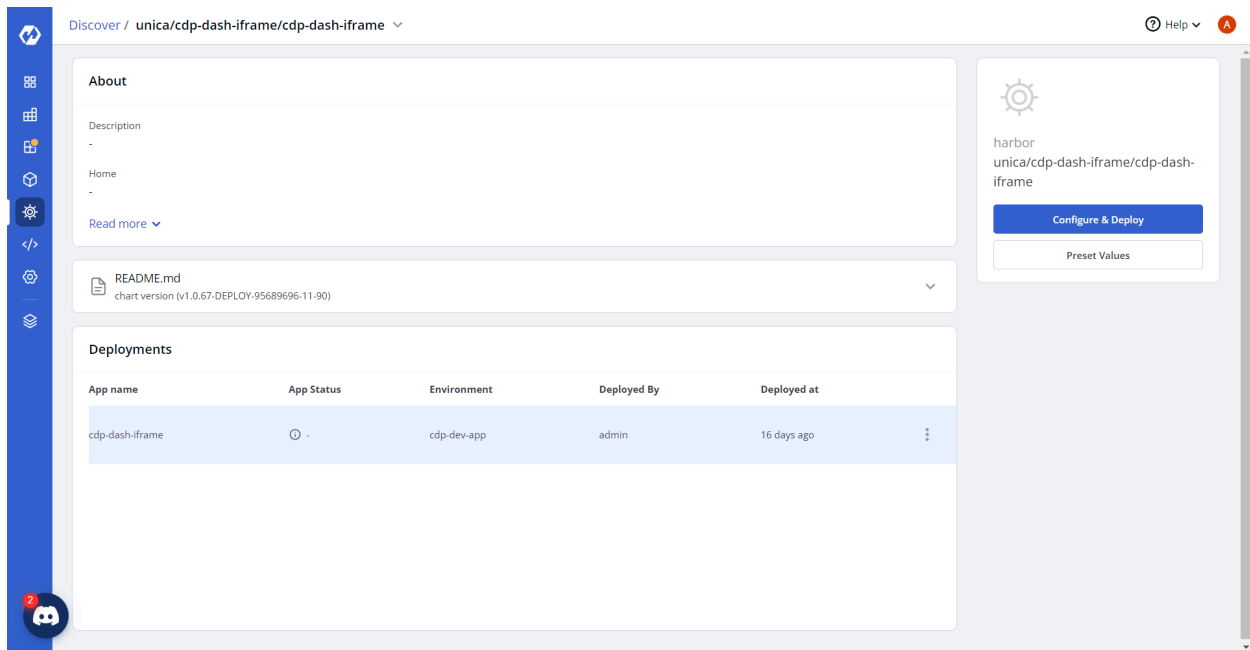
Deploying CDP Dash IFrame

To deploy the CDP Dash IFrame, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-dash-iframe chart to deploy.



2. Now, configure and deploy the CDP Dash IFrame charts.



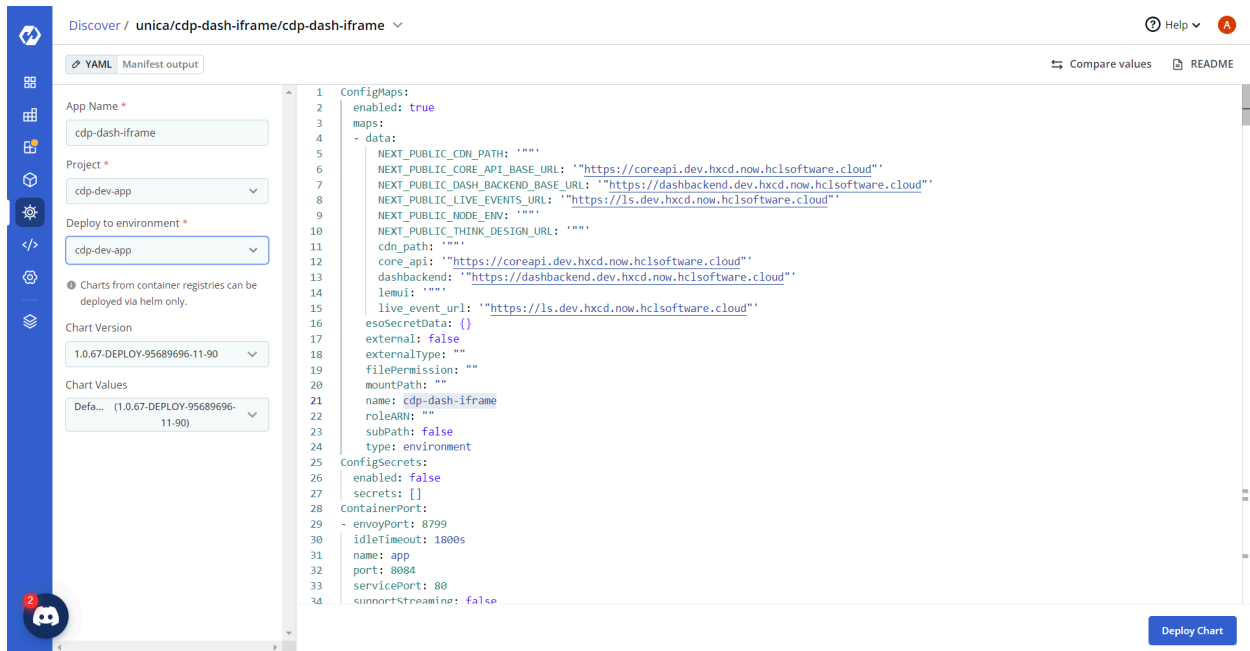
3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
NEXT_PUBLIC_CDN_PATH: ''
NEXT_PUBLIC_CORE_API_BASE_URL: <CORE_API_BASE_URL>
NEXT_PUBLIC_DASH_BACKEND_BASE_URL: <DASH_BACKEND_BASE_URL>
NEXT_PUBLIC_LIVE_EVENTS_URL: <LIVE_EVENTS_URL>
NEXT_PUBLIC_NODE_ENV: ''
NEXT_PUBLIC_THINK_DESIGN_URL: ''
```

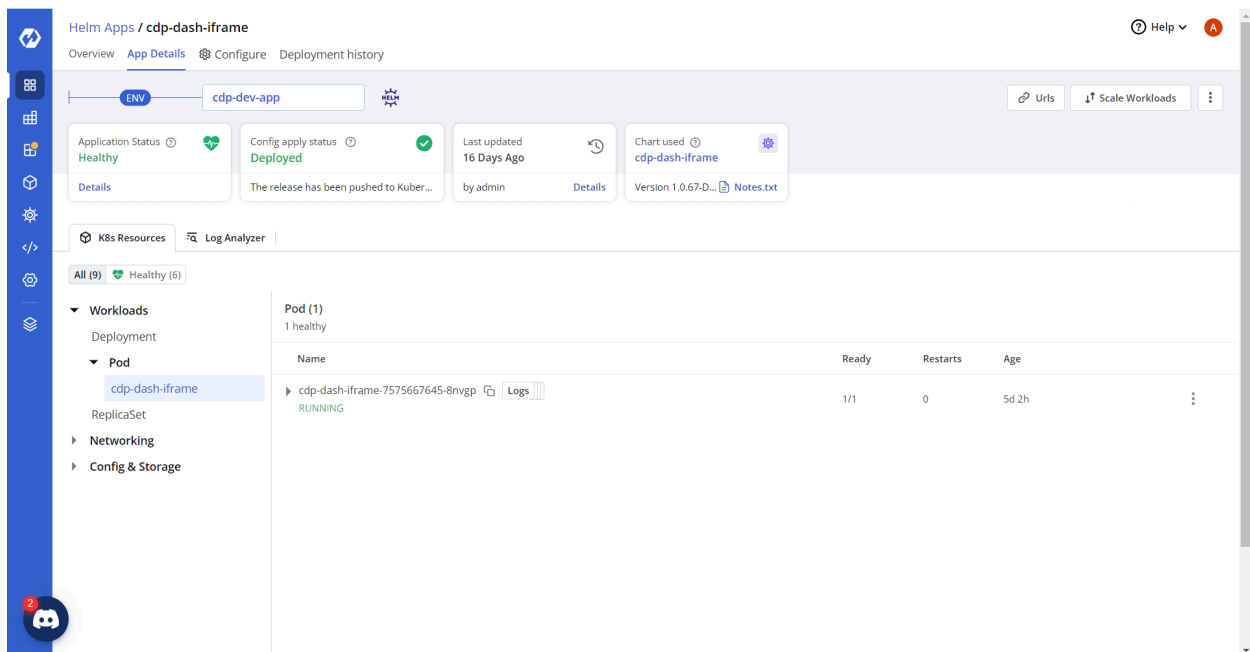
```

cdn_path: ''
core_api:<core_api_url>
dashbackend:<dashbackend_url>
lemui: ''
live_event_url: <live_event_url>

```



4. On successful deployment, validate the deployment as shown below.

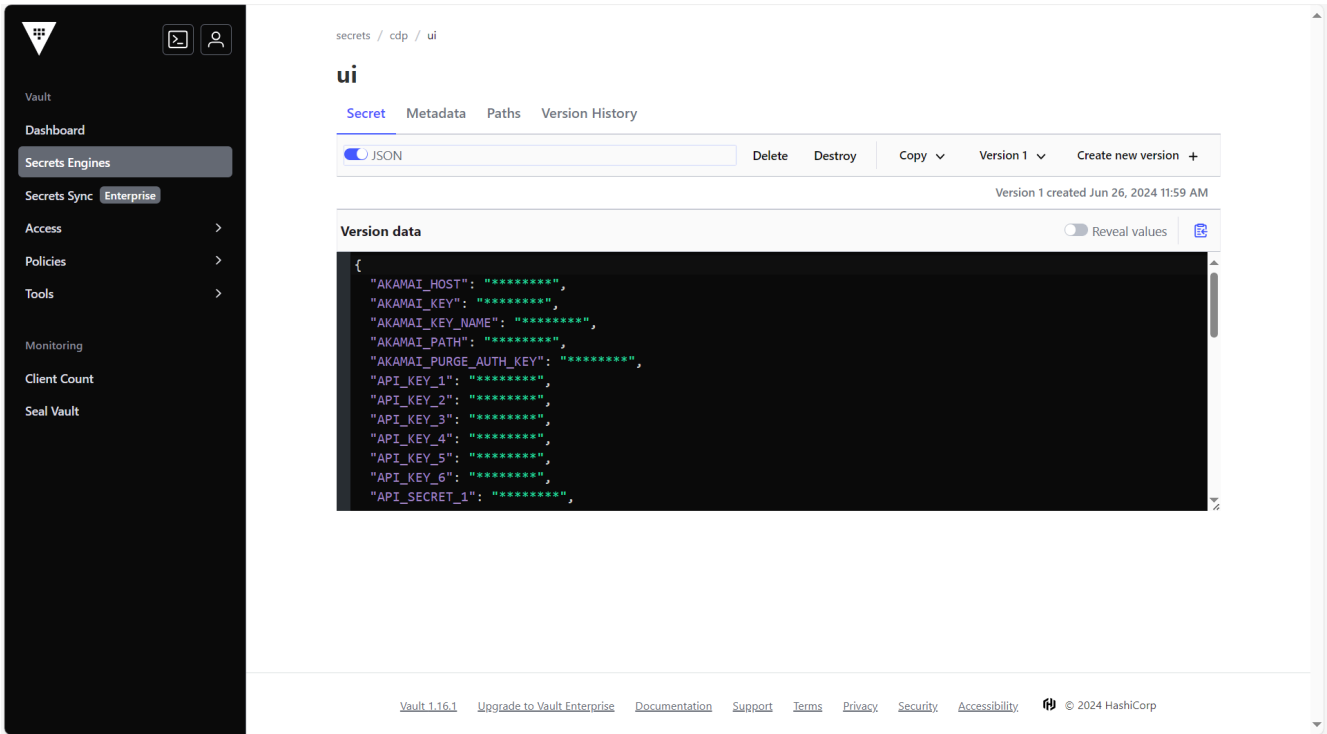


Deploying CDP Web App

This section provides detailed instructions on how to deploy HCL CDP Web App using the Devtron in the OpenShift.

Prerequisites:

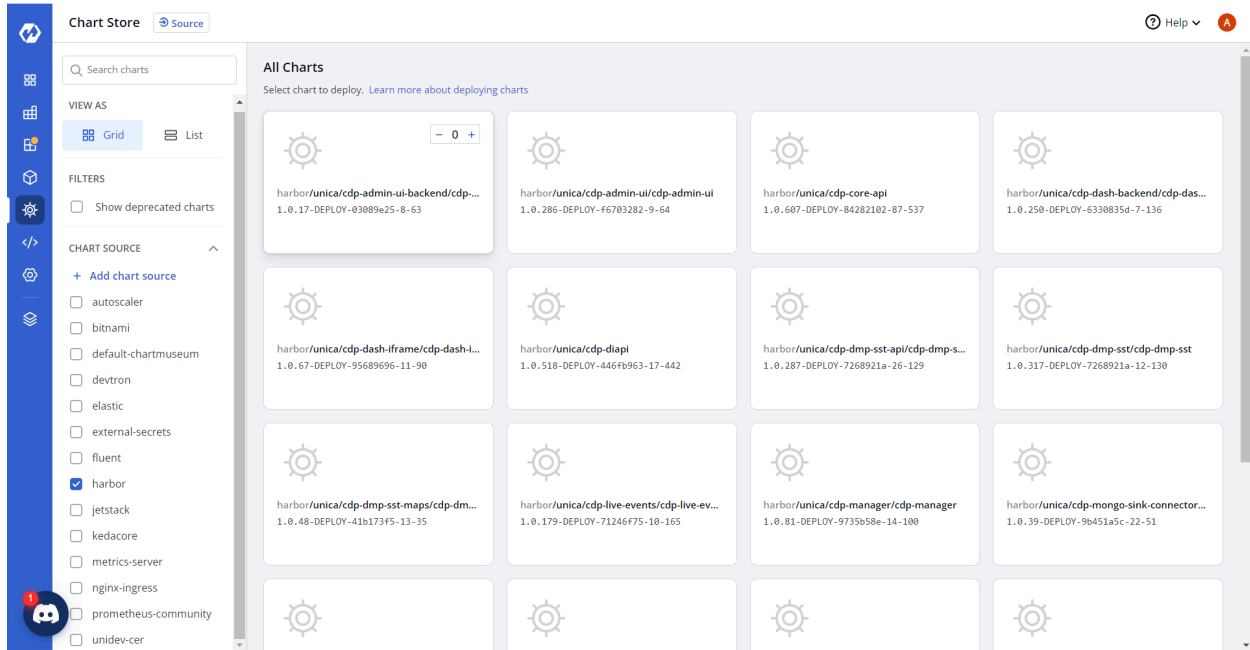
Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Web App.



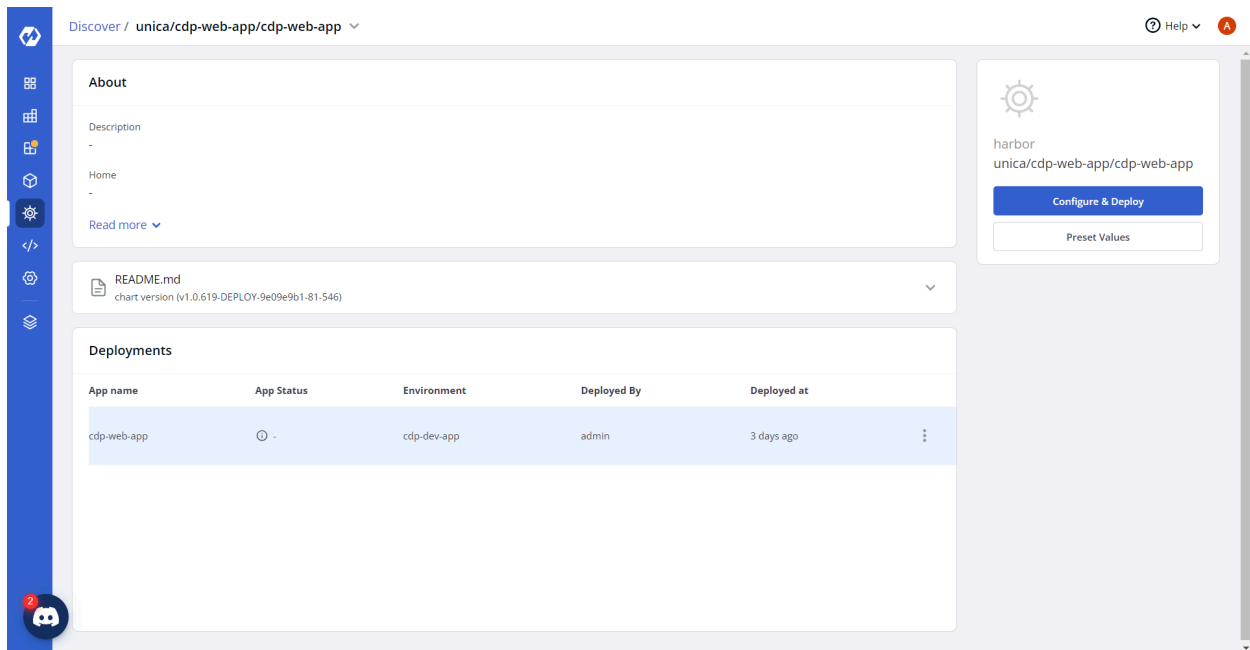
Deploying CDP Web App

To deploy the CDP Web App, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-web-app chart to deploy.



2. Now, configure and deploy the CDP Web App charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
APP_URL: <WebApp_URL>
NEXT_PUBLIC_CDN_PATH: ''
NEXT_PUBLIC_CHANNELS_FLOW_CALL_CENTER_API: <dash_iframe_url>/app/#/channels/call-center-api
NEXT_PUBLIC_CHANNELS_FLOW_EMAIL_API: <dash_iframe_url>/app/#/channels/email-api
NEXT_PUBLIC_CHANNELS_FLOW_EXTERNAL_API: <dash_iframe_url>/app/#/channels/external-api
NEXT_PUBLIC_CHANNELS_FLOW_FACEBOOK_EXPORT: <dash_iframe_url>/app/#/channels/facebook-export
```

```

NEXT_PUBLIC_CHANNELS_FLOW_GOOGLE_EXPORT: <dash_iframe_url>/app/#/channels/google-export
NEXT_PUBLIC_CHANNELS_FLOW_NOTBOT: <dash_iframe_url>/app/#/channels/notification-bot
NEXT_PUBLIC_CHANNELS_FLOW_ONSITE: <dash_iframe_url>/app/#/channels/onsite-notification
NEXT_PUBLIC_CHANNELS_FLOW_QUOTE_API: <dash_iframe_url>/app/#/channels/quote-api
NEXT_PUBLIC_CHANNELS_FLOW_SMS_API: <dash_iframe_url>/app/#/channels/sms-api
NEXT_PUBLIC_COOKIE_DOMAIN: <COOKIE_DOMAIN>
NEXT_PUBLIC_CORE_API_BASE_URL: <CORE_API_BASE_URL>
NEXT_PUBLIC_CUSTOMER_PROPERTIES: <dash_iframe_url>/app/#/customerProperties/list
NEXT_PUBLIC_DASH_BACKEND_BASE_URL: <DASH_BACKEND_BASE_URL>
NEXT_PUBLIC_HOME_ROUTE: /app/cdpDashboard
NEXT_PUBLIC_JOURNEY_LIST: <dash_iframe_url>/app/#/journeyList/
NEXT_PUBLIC_LIVE_EVENTS_URL: <LIVE_EVENTS_URL>
NEXT_PUBLIC_MIXPANEL_KEY: <MIXPANEL_KEY>
NEXT_PUBLIC_NODE_ENV: ''prod''
NEXT_PUBLIC_OFFLINE_DATASOURCE_LIST: <dash_iframe_url>/app/#/offlineDataSource/list
NEXT_PUBLIC_PROFILE_UPLOAD: <dash_iframe_url>/app/#/profileUpload
NEXT_PUBLIC_SEGMENT_CALL_CENTER_API:
  <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/call-center-api
NEXT_PUBLIC_SEGMENT_EMAIL_API: <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/email-api
NEXT_PUBLIC_SEGMENT_EXT_API: <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/external-api
NEXT_PUBLIC_SEGMENT_FB_EXPORT:
  <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/facebook-export
NEXT_PUBLIC_SEGMENT_GOOGLE_EXPORT:
  <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/google-export
NEXT_PUBLIC_SEGMENT_NOT_BOT:
  <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/notification-bot
NEXT_PUBLIC_SEGMENT_OSN:
  <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/onsite-notification
NEXT_PUBLIC_SEGMENT_QUOTE_API: <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/quote-api
NEXT_PUBLIC_SEGMENT_SMS_API: <dash_iframe_url>/app/#/segments/list/view/:segmentId/channels/sms-api
NEXT_PUBLIC_SENTRY_DSN: ''''
NEXT_PUBLIC_THINK_DESIGN_URL: ''<dash_iframe_url>''
SENTRY_PROJECT: ''us-tkd''
SERVER_NAME: <WebApp_server_name>

```

Discover / unica/cdp-web-app/cdp-web-app

YAML Manifest output

App Name *
cdp-web-app

Project *
cdp-dev-app

Deploy to environment *
cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version
1.0.619-DEPLOY-9e09e9b1-81-546

Chart Values
Def... (1.0.619-DEPLOY-9e09e9b1-81-546)

ConfigMaps:

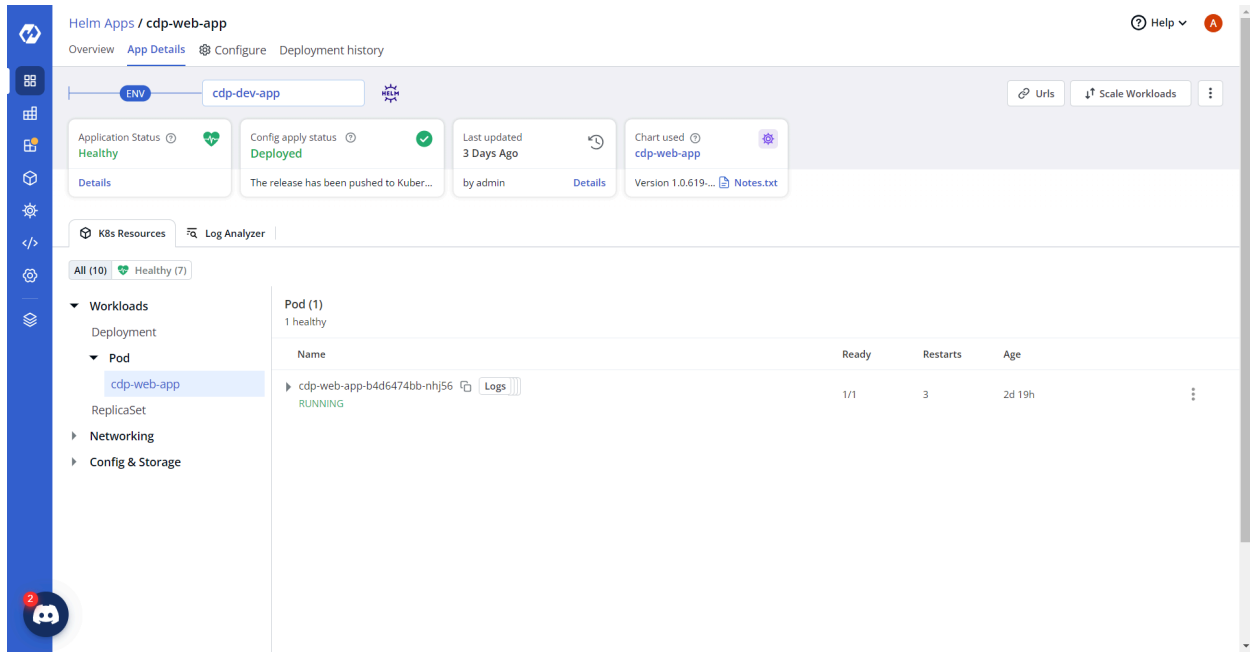
```

1 enabled: true
2 maps:
3   - data:
4     APP_URL: http://cdp-web-app-service-cdp-dev-app.apps.ocp415.manishkr.nonprod.hclpnp.com
5     NEXT_PUBLIC_CDN_PATH: ""
6     NEXT_PUBLIC_CHANNELS_FLOW_CALL_CENTER_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/call-center-api
7     NEXT_PUBLIC_CHANNELS_FLOW_EMAIL_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/email-api
8     NEXT_PUBLIC_CHANNELS_FLOW_EXTERNAL_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/external-api
9     NEXT_PUBLIC_CHANNELS_FLOW_FACEBOOK_EXPORT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/facebook-export
10    NEXT_PUBLIC_CHANNELS_FLOW_GOOGLE_EXPORT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/google-export
11    NEXT_PUBLIC_CHANNELS_FLOW_NOTBOT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/notification-bot
12    NEXT_PUBLIC_CHANNELS_FLOW_ONSITE: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/onsite-notification
13    NEXT_PUBLIC_CHANNELS_FLOW_QUOTE_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/quote-api
14    NEXT_PUBLIC_CHANNELS_FLOW_SMS_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/channels/sms-api
15    NEXT_PUBLIC_COOKIE_DOMAIN: "apps.ocp415.manishkr.nonprod.hclpnp.com"
16    NEXT_PUBLIC_CORE_API_BASE_URL: "http://cdp-core-api-hbr-cdp-core-api-service.cdp-dev-app/"
17    NEXT_PUBLIC_CUSTOMER_PROPERTIES: http://cdp-dash-iframe-service.cdp-dev-app/app/#/customerProperties/list
18    NEXT_PUBLIC_DASH_BACKEND_BASE_URL: "http://cdp-dash-backend-service.cdp-dev-app/"
19    NEXT_PUBLIC_HOME_ROUTE: /app/cdpDashboard
20    NEXT_PUBLIC_JOURNEY_LIST: http://cdp-dash-iframe-service.cdp-dev-app/app/#/journeyList/
21    NEXT_PUBLIC_LIVE_EVENTS_URL: "https://ls.dev.hxcd.now.hclsoftware.cloud/"
22    NEXT_PUBLIC_MIXPANEL_KEY: "e361c5a0413b113417b2f9cd576cee01"
23    NEXT_PUBLIC_NODE_ENV: "prod"
24    NEXT_PUBLIC_OFFLINE_DATASOURCE: Follow link (ctrl + click) lash-iframe-service.cdp-dev-app/app/#/offlineDataSource/list
25    NEXT_PUBLIC_PROFILE_UPLOAD: http://cdp-dash-iframe-service.cdp-dev-app/app/#/profileUpload
26    NEXT_PUBLIC_SEGMENT_CALL_CENTER_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/call-center-api
27    NEXT_PUBLIC_SEGMENT_EMAIL_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/email-api
28    NEXT_PUBLIC_SEGMENT_EXT_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/external-api
29    NEXT_PUBLIC_SEGMENT_FB_EXPORT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/facebook-export
30    NEXT_PUBLIC_SEGMENT_GOOGLE_EXPORT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/google-export
31    NEXT_PUBLIC_SEGMENT_NOT_BOT: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/notification-bot
32    NEXT_PUBLIC_SEGMENT_OSN: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/onsite-notification
33    NEXT_PUBLIC_SEGMENT_QUOTE_API: http://cdp-dash-iframe-service.cdp-dev-app/app/#/segments/list/view/:segmentId/channels/quote-api
34

```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

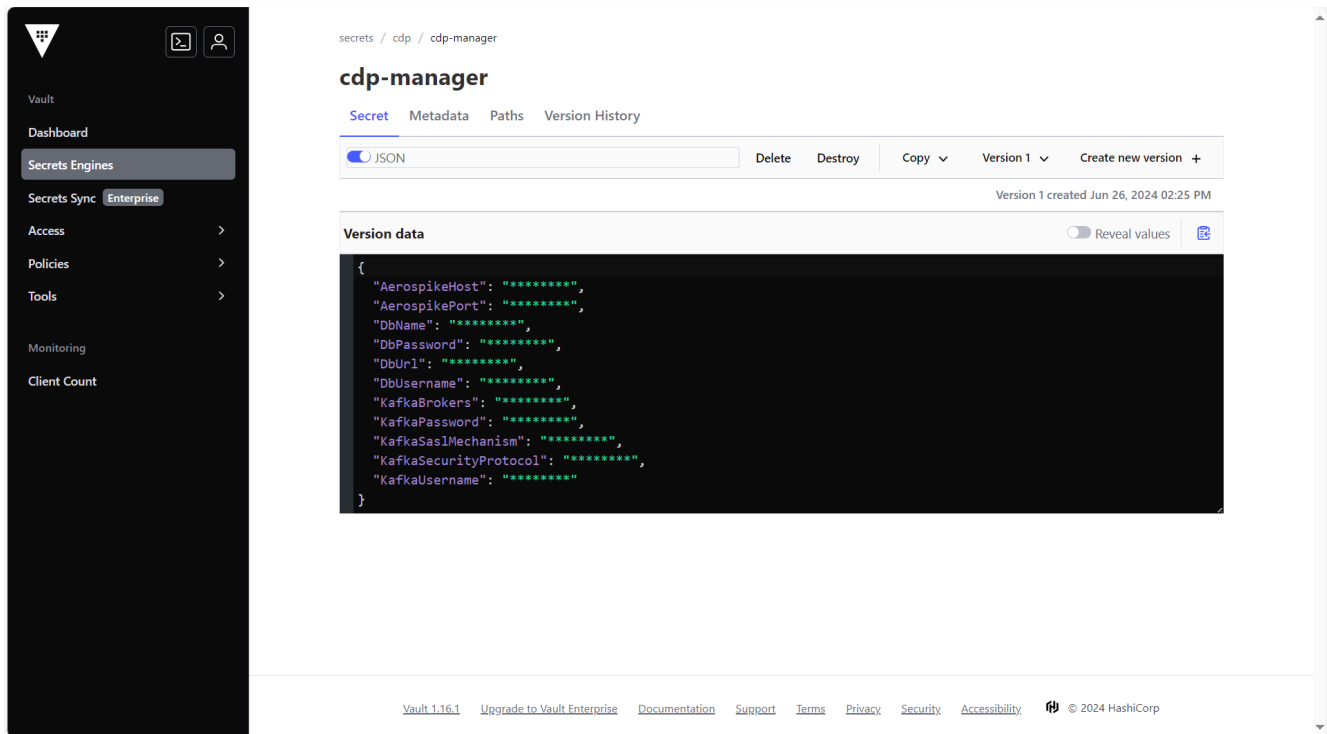


Deploying CDP Manager

This section provides detailed instructions on how to deploy HCL CDP Manager using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Manager.



To create the UI secret in HashiCorp Vault, follow the steps below:

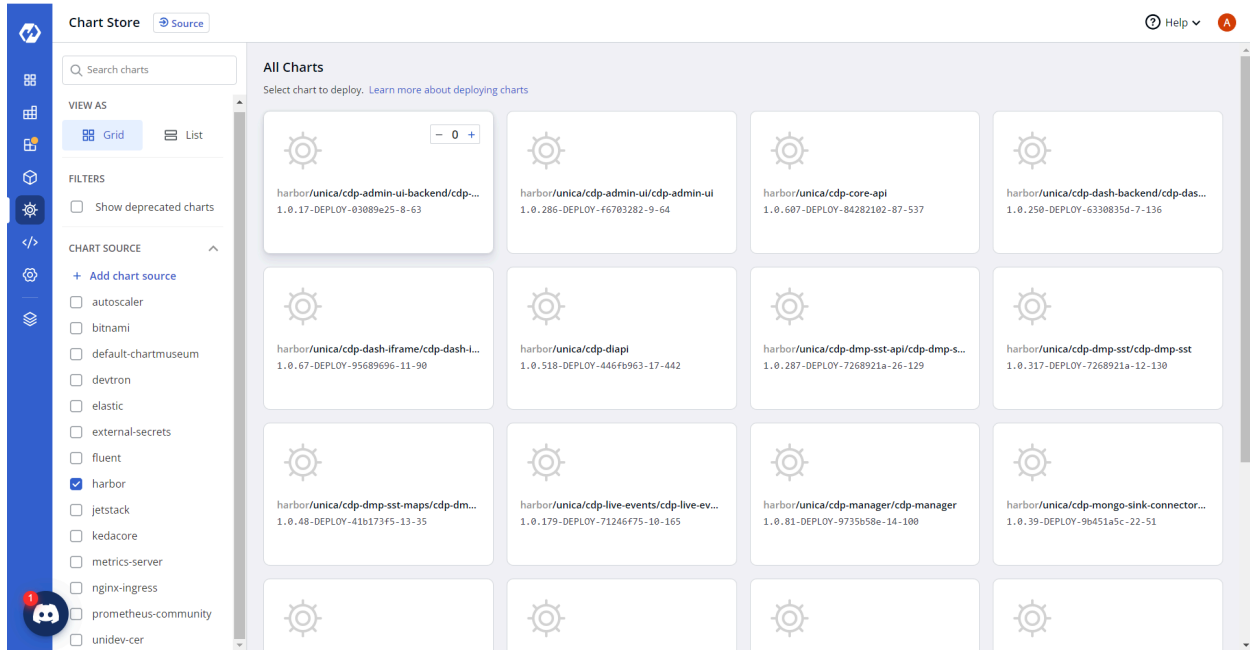
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "AerospikeHost": "<AerospikeHost>",
  "AerospikePort": "<AerospikePort>",
  "DbName": "<DbName>",
  "DbPassword": "<DbPassword>",
  "DbUrl": "<ip:port>",
  "DbUsername": "<DbUsername>",
  "KafkaBrokers": "<KafkaBrokers>",
  "KafkaPassword": "<KafkaPassword>",
  "KafkaSaslMechanism": "<KafkaSaslMechanism>",
  "KafkaSecurityProtocol": "<KafkaSecurityProtocol>",
  "KafkaUsername": "<KafkaUsername>"
}
```

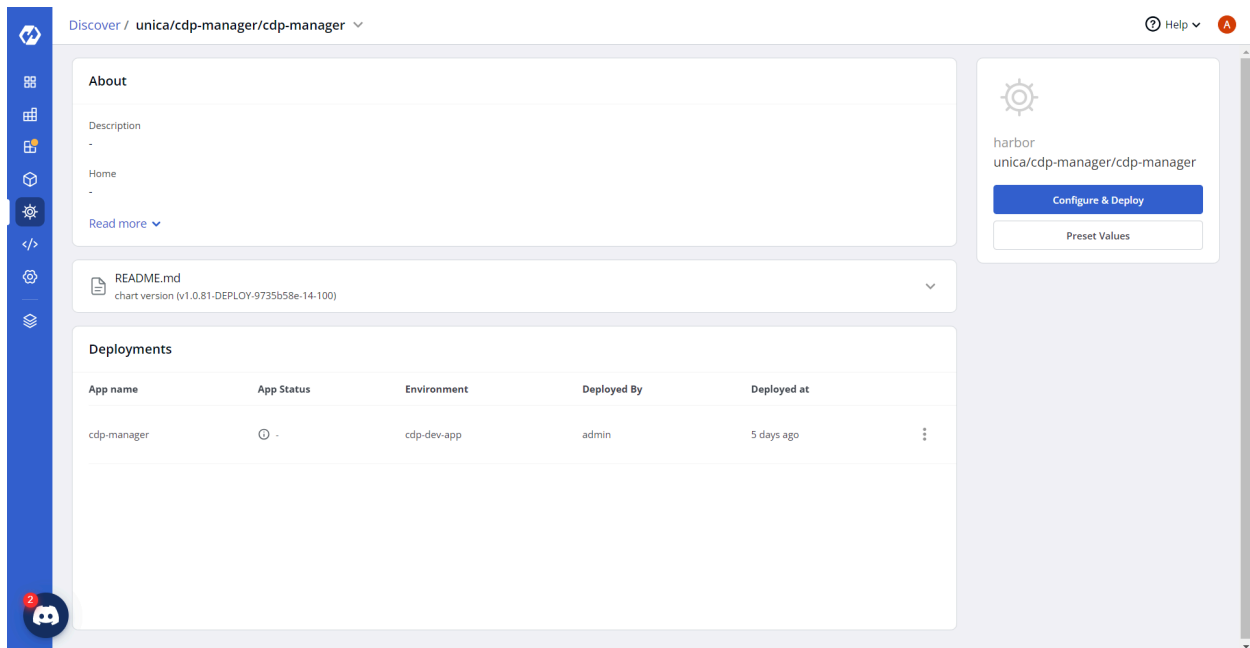
Deploying CDP Manager

To deploy the CDP Manager, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-manager chart to deploy.



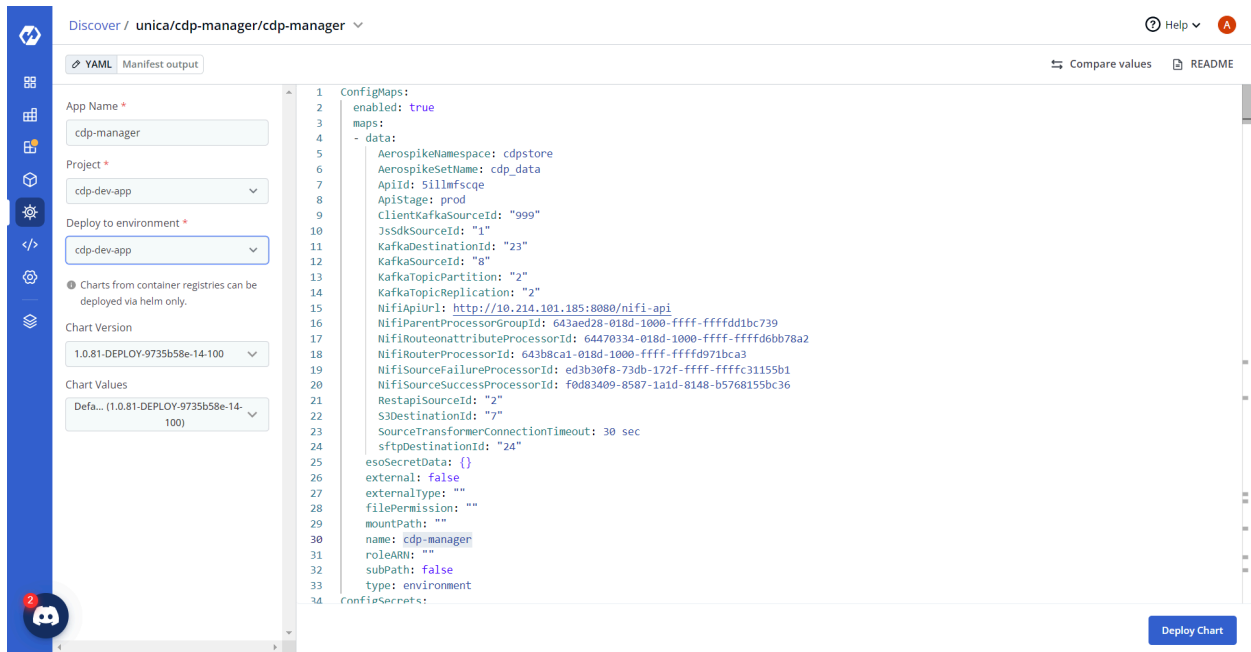
2. Now, configure and deploy the CDP Manager charts.



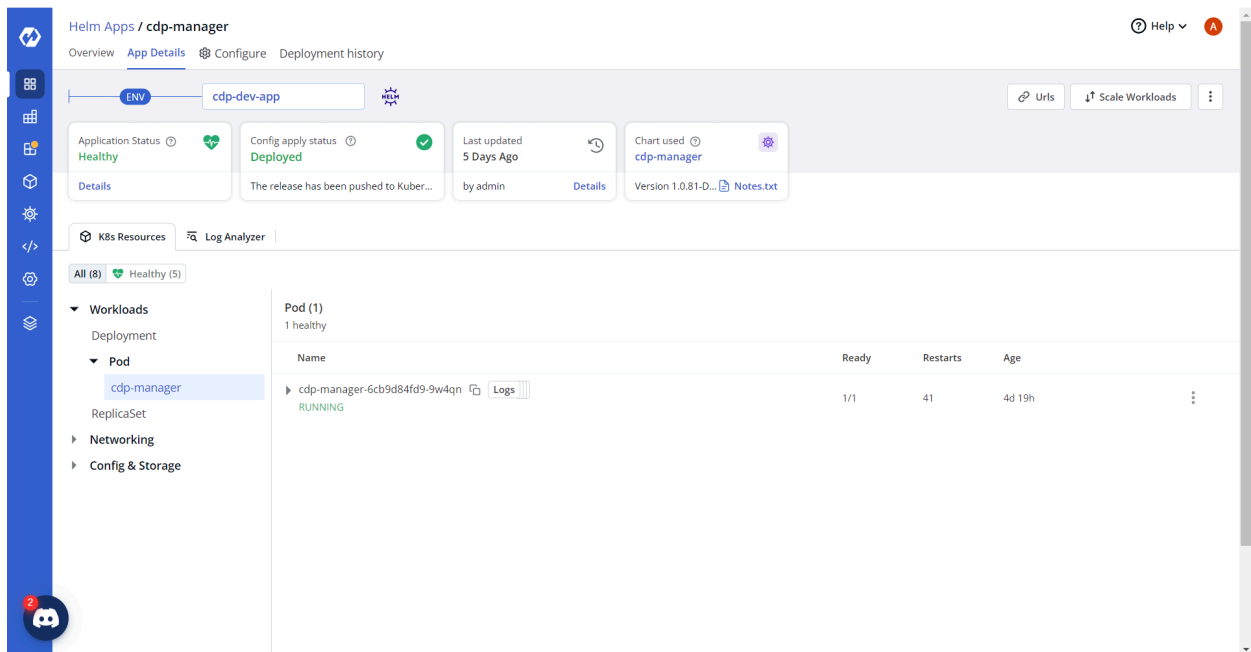
3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
AerospikeNamespace: <AerospikeNamespace>
AerospikeSetName: <AerospikeSetName>
NifiApiUrl: http://<ip:port>/nifi-api
NifiParentProcessorGroupId: <NifiParentProcessorGroupId>
NifiRouteonattributeProcessorId: <NifiRouteonattributeProcessorId>
NifiRouterProcessorId: <NifiRouterProcessorId>
```

```
NifiSourceFailureProcessorId: <NifiSourceFailureProcessorId>
NifiSourceSuccessProcessorId: <NifiSourceSuccessProcessorId>
```



4. On successful deployment, validate the deployment as shown below.

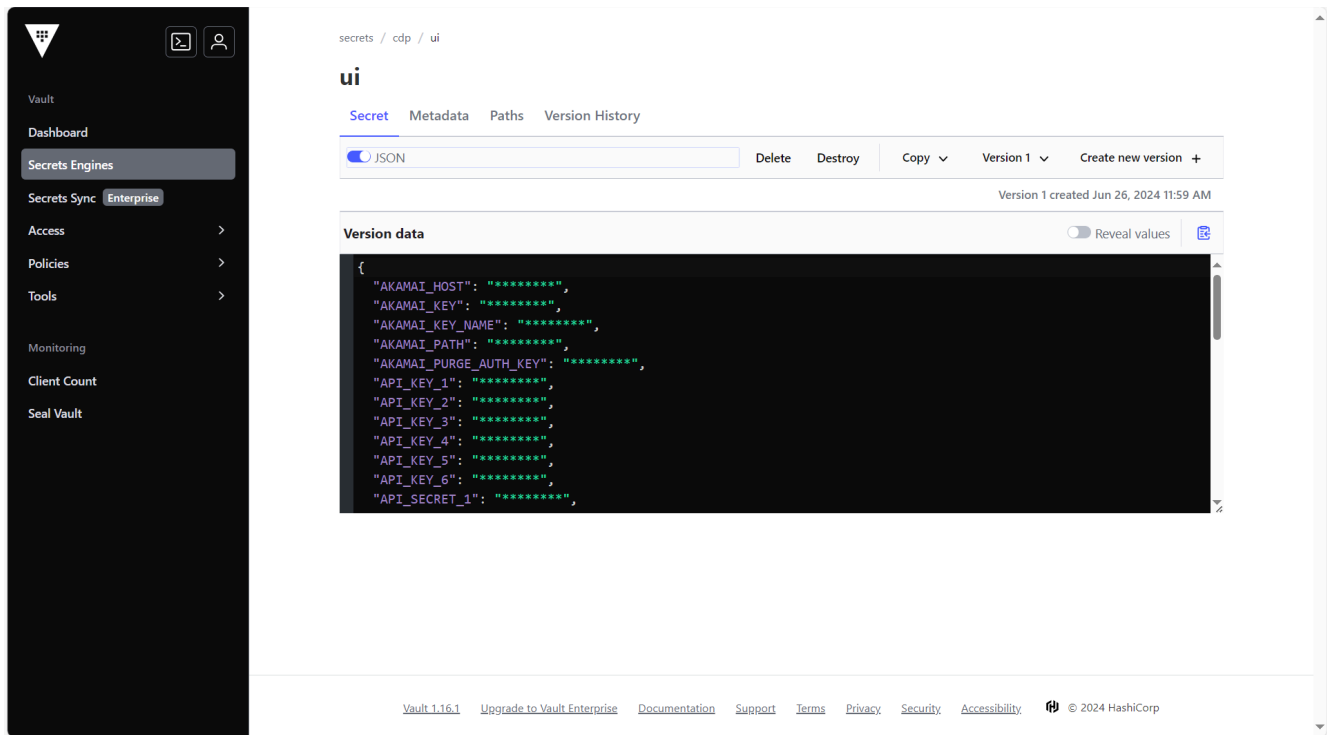


Deploying CDP Pixel Listener

This section provides detailed instructions on how to deploy HCL CDP Pixel Listener using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Pixel Listener.



To create the UI secret in HashiCorp Vault, follow the steps below:

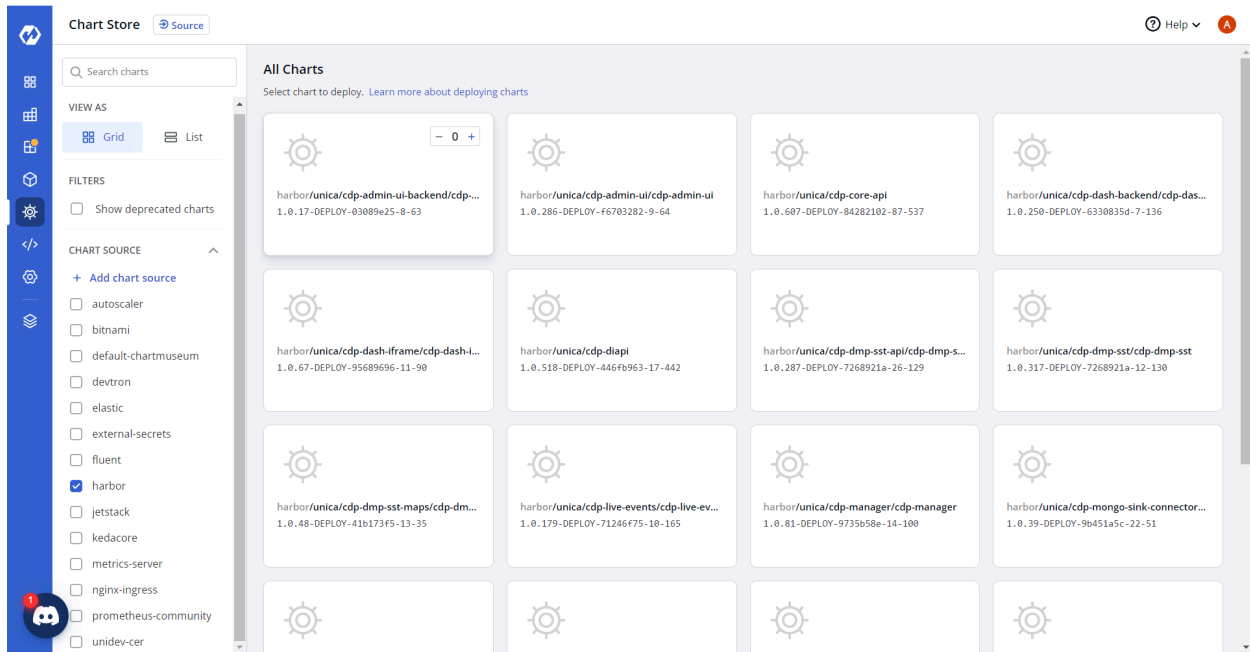
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "ADCB_AERO_CLUSTER": "<Aerospike_ip:port>",
  "KAFKA_PRODUCER_CONFIG": "<KAFKA_PRODUCER_CONFIG>",
  "PII_AERO_APS1_HOST": "<Aerospike_ip>",
  "PII_AERO_APS1_PORT": "<Aerospike_port>",
  "PII_AERO_HOST": "<Aerospike_ip>",
  "PII_AERO_PORT": "<Aerospike_port>",
  "PII_AERO_USE1_HOST": "<Aerospike_ip>",
  "PII_AERO_USE1_PORT": "<Aerospike_port>",
  "TSDB_HOST": "<TSDB_HOST>",
  "TSDB_PORT": "<TSDB_PORT>",
  "VRM_DB_PASSWORD": "<VRM_DB_PASSWORD>",
  "VRM_DB_URL": "jdbc:mariadb://<ip:port>/<dbName>",
  "VRM_DB_USER": "<VRM_DB_USER>"
}
```

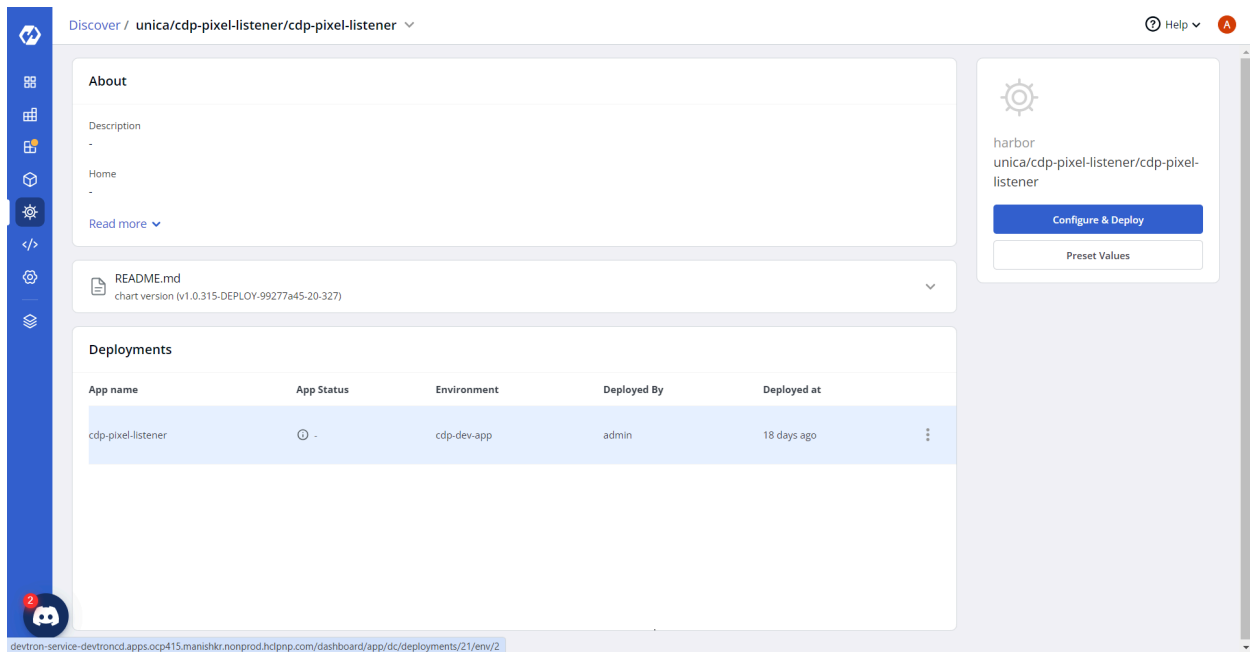
Deploying CDP Pixel Listener

To deploy the CDP Pixel Listener, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-pixel-listener chart to deploy.



2. Now, configure and deploy the CDP Pixel Listener charts.



3. In the **YAML** section, update the ConfigMap using below details and deploy the charts.

```
ALLOWED_CAMPAIGNS: all
CDN_DOMAIN: <CDN_DOMAIN>
COOKIE_DOMAIN: <COOKIE_DOMAIN>
DB_DRIVER: org.mariadb.jdbc.Driver
DEFAULT_DOMAIN: <DEFAULT_DOMAIN>
```


The screenshot shows the Helm Charts UI for the `cdp-pixel-listener` chart. The left sidebar contains navigation icons. The main area displays the chart's configuration, including the App Name, Project, and Deploy to environment. The right pane shows the `values.yaml` file with various configuration parameters.

```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5         ADD_MESSAGE_ID_CAMPAIGNS: 6081,6108
6         AERO_RECORD_TTL: "10"
7         ALLOWED_CAMPAIGNS: all
8         BLACK_LISTED_MOBILE_VIZID: 00000000-0000-0000-0000-000000000000
9         CDN_DOMAIN: cdn8.dev.hxcd.now.hclsoftware.cloud
10        COOKIE_DOMAIN: .hclsoftware.cloud
11        DB_DRIVER: org.mariadb.jdbc.Driver
12        DEFAULT_DOMAIN: dev.hxcd.now.hclsoftware.cloud
13        ENTRY_FILTER_CONFIG: VIZVRM1422|true|||100\nVIZ-VRM-1200||t100||100\nVIZVRM285||e100||90\nVIZVRM3128|||100\nVIZVRM2963||e100|1|1|100\nV
14        EXECUTOR_SERVICE_MAX_THREADS: "32"
15        EXECUTOR_SERVICE_MIN_THREADS: "32"
16        EXECUTOR_SERVICE_QUEUE_SIZE: "10000"
17        EXECUTOR_SERVICE_THREAD_TTL: "5"
18        FAULTY_COOKIE: viz_551bd527a93ec
19        FAULTY_COOKIE_LIST: viz_551bd527a93ec,viz_551bd527a9fff
20        FAULTY_COOKIE_TRANSFORM_THRESHOLD: "10"
21        GET_ANALYZE_CAMPAIGN_IDS: VIZVRM6081,VIZVRM6106,VIZVRM6107,VIZVRM6112,VIZVRM6142,VIZVRM6153,VIZVRM6154
22        GET_HASH_ENABLED: "true"
23        GET_KMS_ENCRYPTION_ENABLED: "true"
24        GET_PARAMS_TO_BE_KMS_ENCRYPTED: name,firstname,lastname
25        GET_PARAMS_TO_HASH: fp55:EMAIL,fp56:MOBILE
26        GET_RAW_HASH_PARAMS_MAP: fp40:fp55,fp41:fp56
27        KAFKA_CONFIG_PROPERTY_KEY_VALUE_SEPARATOR: ':'
28        KAFKA_PRODUCER_EVENT_TYPE: "0"
29        KAFKA_PRODUCER_HOSTNAME: pixel-listener-mu
30        KAFKA_PRODUCER_META: "false"
31        KAFKA_PRODUCER_POOL_SIZE: "10"
32        KMS_CAMPAIGN_DEFAULT_TTL: "21600"
33        KMS_LAZY_LOAD_ENABLED: "false"
34        NEW_FAULTY_COOKIE: viz_551bd527a9fff
  
```

At the bottom right, there is a **Deploy Chart** button.

4. On successful deployment, validate the deployment as shown below.

The screenshot shows the Helm Apps UI for the `cdp-pixel-listener` chart. The left sidebar contains navigation icons. The main area displays the chart's deployment status, including the Application Status, Config apply status, Last updated, and Chart used. The right pane shows the deployment details, including the Pod (1) and its status.

Application Status: Healthy

Config apply status: Deployed

Last updated: 18 Days Ago

Chart used: cdp-pixel-listener

Deployment Details:

Name	Ready	Restarts	Age
cdp-pixel-listener-848b7f68c5-pphth	1/1	0	5d 2h

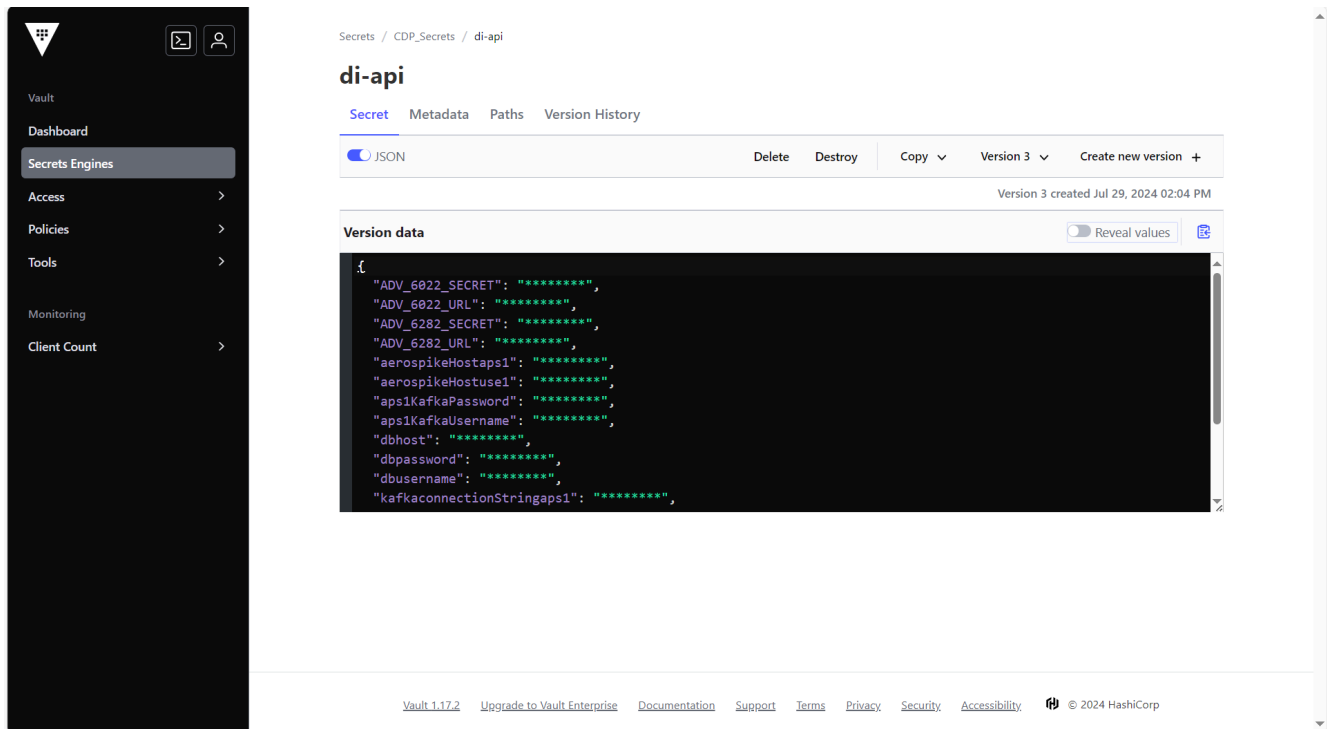
The pod is in the **Running** state.

Deploying CDP DI API

This section provides detailed instructions on how to deploy HCL CDP DI API using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP DI API.



To create the UI secret in HashiCorp Vault, follow the steps below:

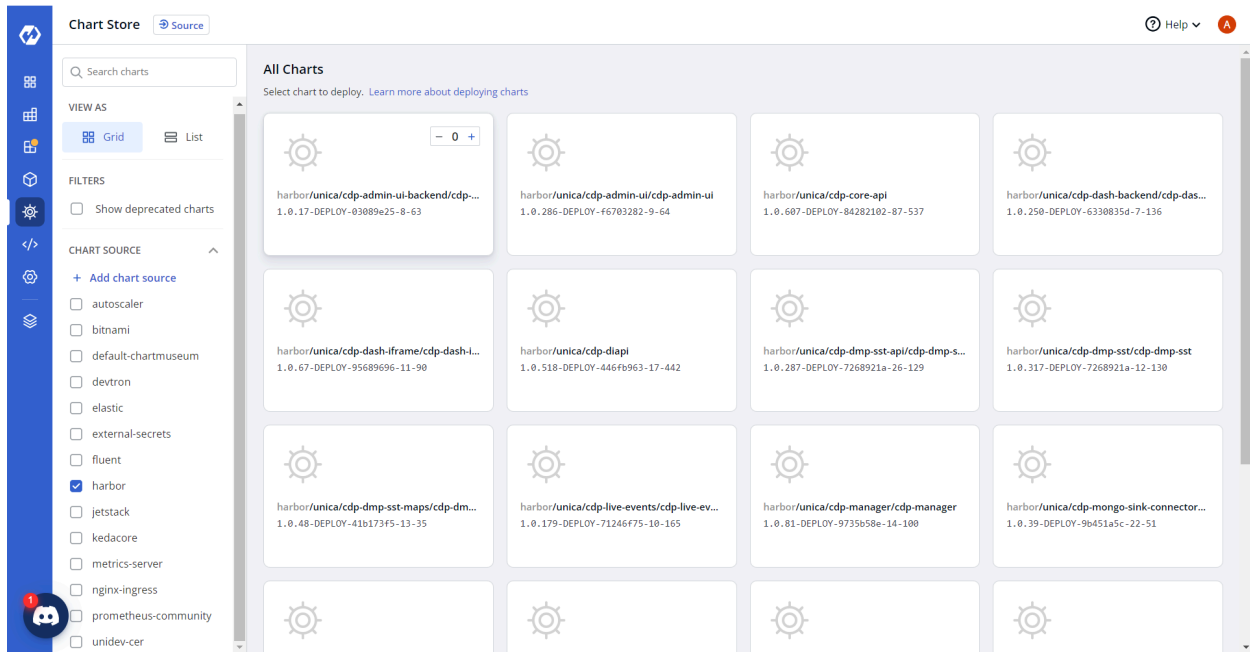
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "ADV_6022_SECRET": "\\\"\\\",
  "ADV_6022_URL": "\\\"\\\",
  "ADV_6282_SECRET": "\\\"\\\",
  "ADV_6282_URL": "\\\"\\\",
  "aerospikeHostaps1": "\\\"<aerospikeHostaps1>\\\",
  "aerospikeHostuse1": "\\\"\\\",
  "aps1KafkaPassword": "\\\"<aps1KafkaPassword>\\\",
  "aps1KafkaUsername": "\\\"<aps1KafkaUsername>\\\",
  "dbhost": "\\\"<dbhost>\\\",
  "dbpassword": "\\\"<dbpassword>\\\",
  "dbusername": "\\\"<dbusername>\\\",
  "kafkaconnectionStringaps1": "\\\"<kafkaconnectionStringaps1>\\\",
  "kafkaconnectionStringuse1": "\\\"\\\",
  "kmsUrl": "\\\"<kmsUrl>\\\",
  "tdsbHost": "\\\"<tdsbHost>\\\",
  "use1KafkaPassword": "\\\"\\\",
  "use1KafkaUsername": "\\\"\\\"
}
```

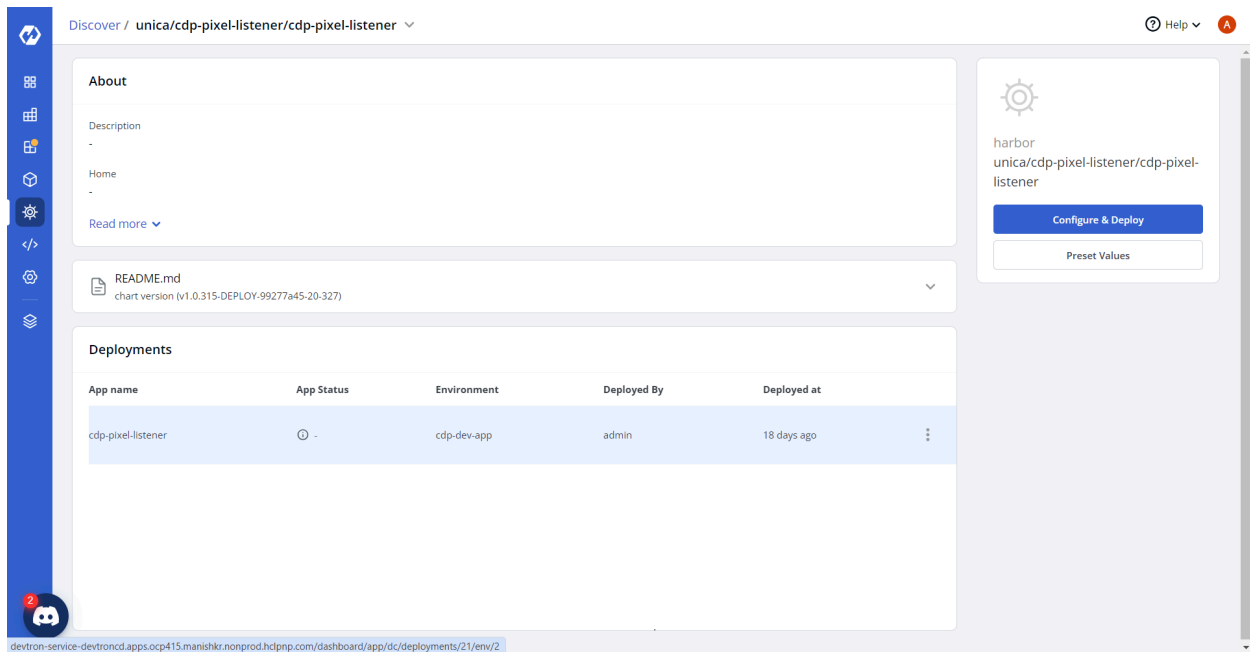
Deploying CDP DI API

To deploy the CDP DI API, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-di-api chart to deploy.



2. Now, configure and deploy the CDP DI API charts.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
aerospikenamespaceaps1: cdpstore
aerospikenamespaceuse1: cdpstore
aps1KafkaAuthenticationTimeout: "5000"
aps1KafkaClientId: DMPDiapiProducer
aps1KafkaConnectionTimeout: "3000"
aps1KafkaReauthenticationThreshold: "10000"
```

```

aps1KafkaSslEnabled: "false"
cdpSenderTopic: cdp_fb_conv_api
database: vrm
dbconnectionLimit: "1"
dbport: "3306"
fromEmail: ""
isEncryptionEnabled: "true"
kafkaProducerEventType: "104"
kafkaRegions: '{"AP_SOUTH_1": "aps1"}'
kafkaSaslMechanism: SCRAM-SHA-512
kafkasource: dataingestionapi
kafkatopic: dmp_sst_nba_di_api
nodeProcessMemoryLimit: 1500M
toEmail: ""
use1KafkaAuthenticationTimeout: "5000"
use1KafkaClientId: DMPDiapiProducer
use1KafkaConnectionTimeout: "3000"
use1KafkaReauthenticationThreshold: "10000"
use1KafkaSslEnabled: "false"

```

The screenshot displays the OpenShift console interface for deploying the `cdp-pixel-listener` chart. The left sidebar contains navigation icons, and the top bar shows the breadcrumb `Discover / unica/cdp-pixel-listener/cdp-pixel-listener`. The main panel is divided into two sections: configuration fields on the left and a YAML manifest on the right.

Configuration Fields:

- App Name:** `cdp-pixel-listener`
- Project:** `cdp-dev-app`
- Deploy to environment:** `cdp-dev-app`
- Chart Version:** `1.0.315-DEPLOY-99277a45-20-327`
- Chart Values:** `Def... (1.0.315-DEPLOY-99277a45-20-327)`

YAML Manifest:

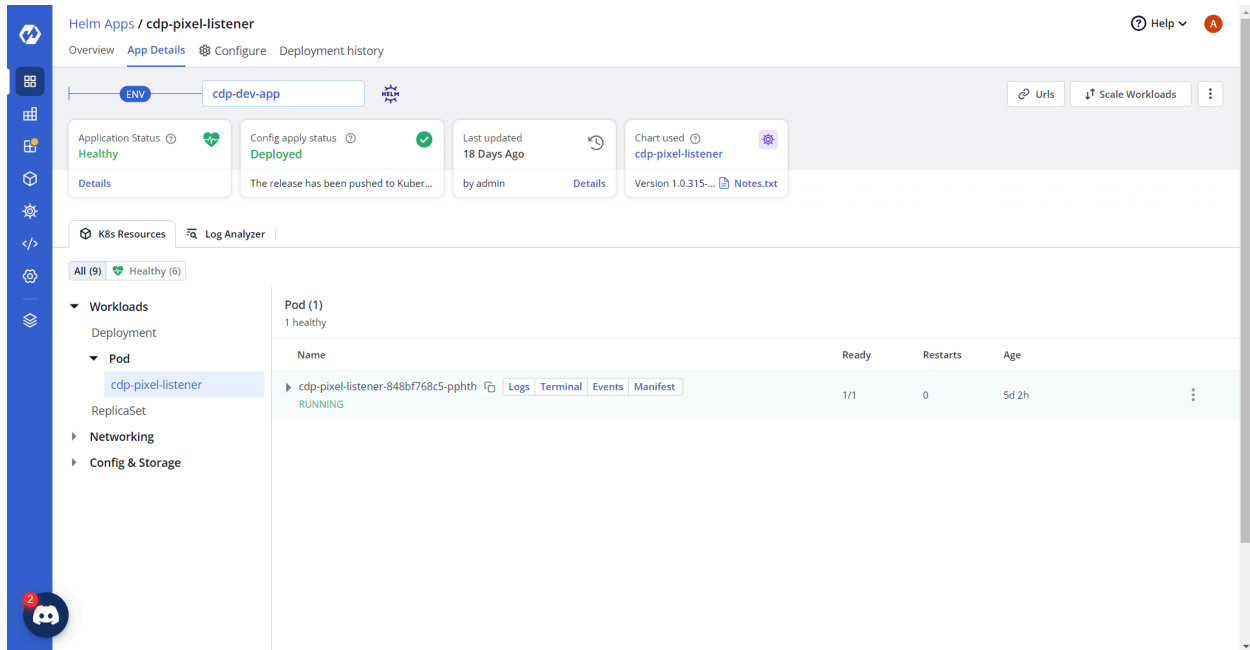
```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5       ADD_MESSAGE_ID_CAMPAIGNS: 6081,6108
6       AERO_RECORD_TTL: "10"
7       ALLOWED_CAMPAIGNS: all
8       BLACK_LISTED_MOBILE_VIZID: 00000000-0000-0000-0000-000000000000
9       CDN_DOMAIN: cdn8.dev.hxcd.now.hclsoftware.cloud
10      COOKIE_DOMAIN: .hclsoftware.cloud
11      DB_DRIVER: org.mariadb.jdbc.Driver
12      DEFAULT_DOMAIN: dev.hxcd.now.hclsoftware.cloud
13      ENTRY_FILTER_CONFIG: VIZVRM1422|true|||100\nVIZ-VRM-1200||t100|||100\nVIZVRM285||e100|||90\nVIZVRM3128|||100\nVIZVRM2963||e100|1|100\nV
14      EXECUTOR_SERVICE_MAX_THREADS: "32"
15      EXECUTOR_SERVICE_MIN_THREADS: "32"
16      EXECUTOR_SERVICE_QUEUE_SIZE: "10000"
17      EXECUTOR_SERVICE_THREAD_TTL: "5"
18      FAULTY_COOKIE: viz_551bd527a93ec
19      FAULTY_COOKIE_LIST: viz_551bd527a93ec,viz_551bd527a9fff
20      FAULTY_COOKIE_TRANSFORM_THRESHOLD: "10"
21      GET_ANALYZE_CAMPAIGN_IDS: VIZVRM6081,VIZVRM6108,VIZVRM6106,VIZVRM6107,VIZVRM6112,VIZVRM6142,VIZVRM6153,VIZVRM6154
22      GET_HASH_ENABLED: "true"
23      GET_KMS_ENCRYPTION_ENABLED: "true"
24      GET_PARAMS_TO_BE_KMS_ENCRYPTED: name,firstname,lastname
25      GET_PARAMS_TO_HASH: fp55:EMAIL,fp56:MOBILE
26      GET_RAW_HASH_PARAMS_MAP: fp40:fp55,fp41:fp56
27      KAFKA_CONFIG_PROPERTY_KEY_VALUE_SEPARATOR: ':'
28      KAFKA_PRODUCER_EVENT_TYPE: "0"
29      KAFKA_PRODUCER_HOSTNAME: pixel-listener-mu
30      KAFKA_PRODUCER_META: "false"
31      KAFKA_PRODUCER_POOL_SIZE: "10"
32      KMS_CAMPAIGN_DEFAULT_TTL: "21600"
33      KMS_LAZY_LOAD_ENABLED: "false"
34      KMS_FAULTY_COOKIE: viz_551bd527a9fff

```

A **Deploy Chart** button is located at the bottom right of the interface.

4. On successful deployment, validate the deployment as shown below.

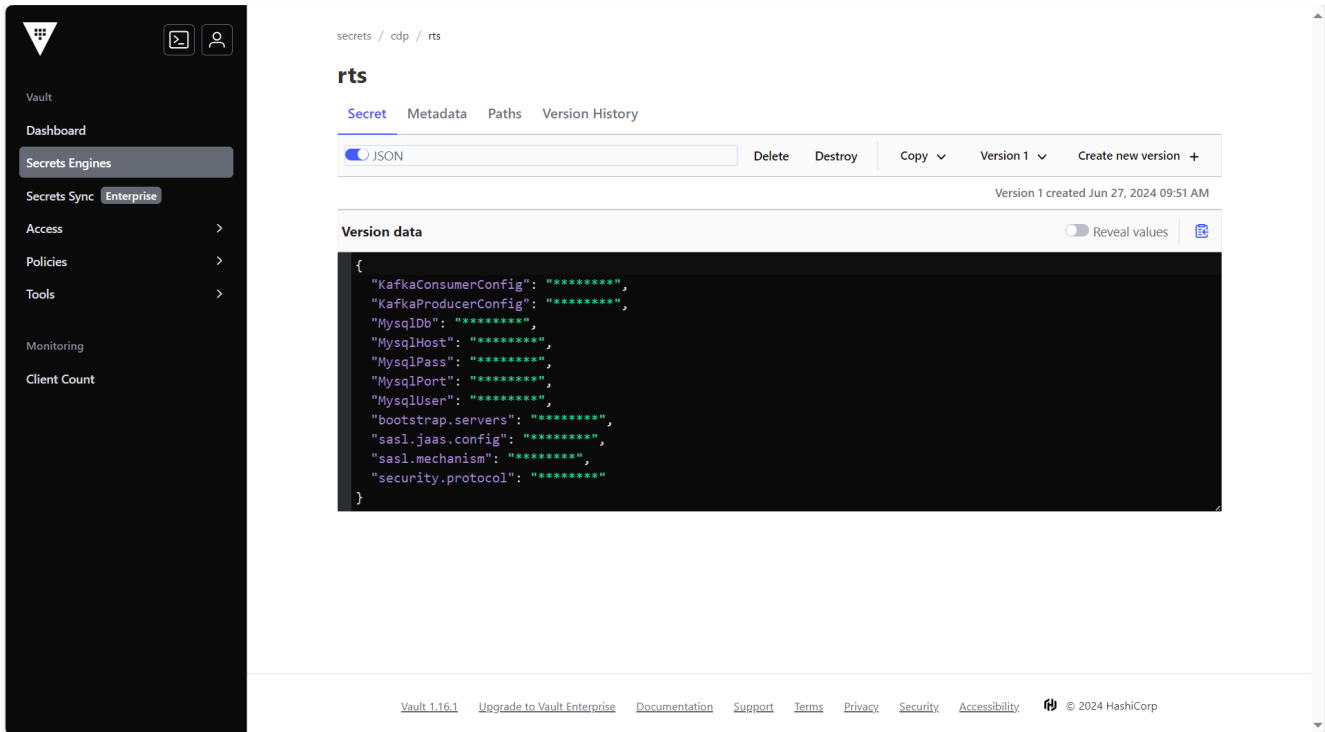


Deploying CDP RTS

This section provides detailed instructions on how to deploy HCL CDP RTS using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP RTS.



To create the UI secret in HashiCorp Vault, follow the steps below:

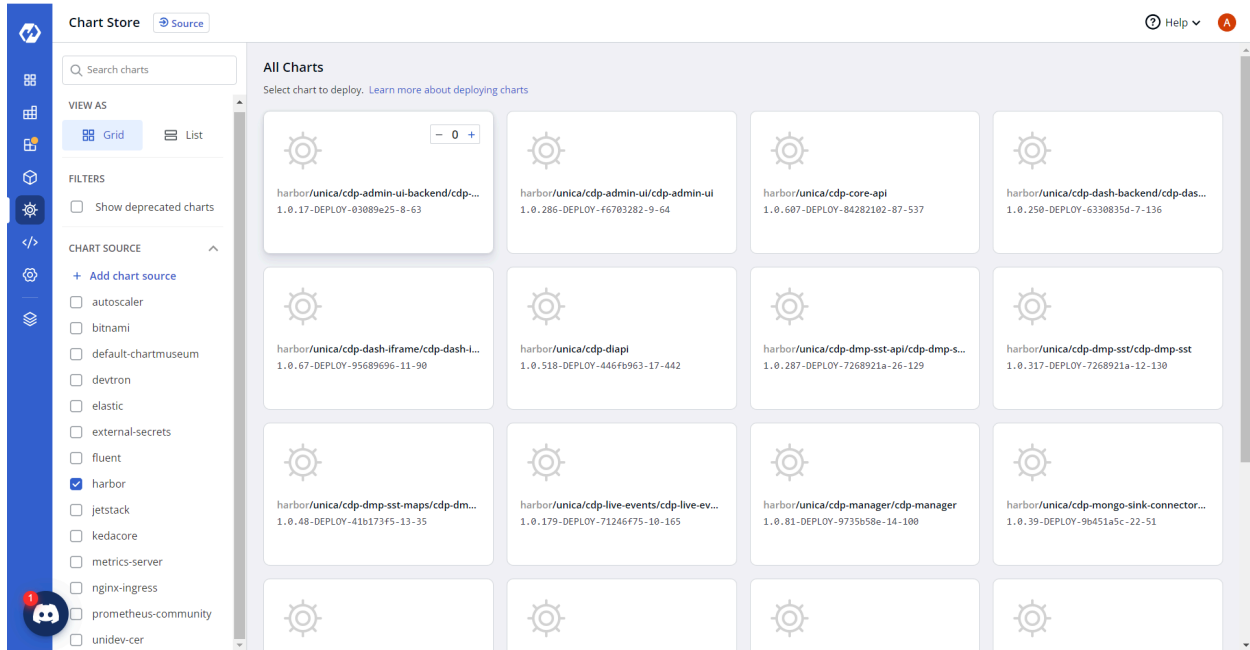
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "KafkaConsumerConfig": "<KafkaConsumerConfig>",
  "KafkaProducerConfig": "<KafkaProducerConfig>",
  "MysqlDb": "<MysqlDb>",
  "MysqlHost": "<MysqlHost>",
  "MysqlPass": "<MysqlPass>",
  "MysqlPort": "<MysqlPort>",
  "MysqlUser": "<MysqlUser>",
  "bootstrap.servers": "<bootstrap.servers>",
  "sas1.mechanism": "<sas1.mechanism>",
  "security.protocol": "<security.protocol>"
}
```

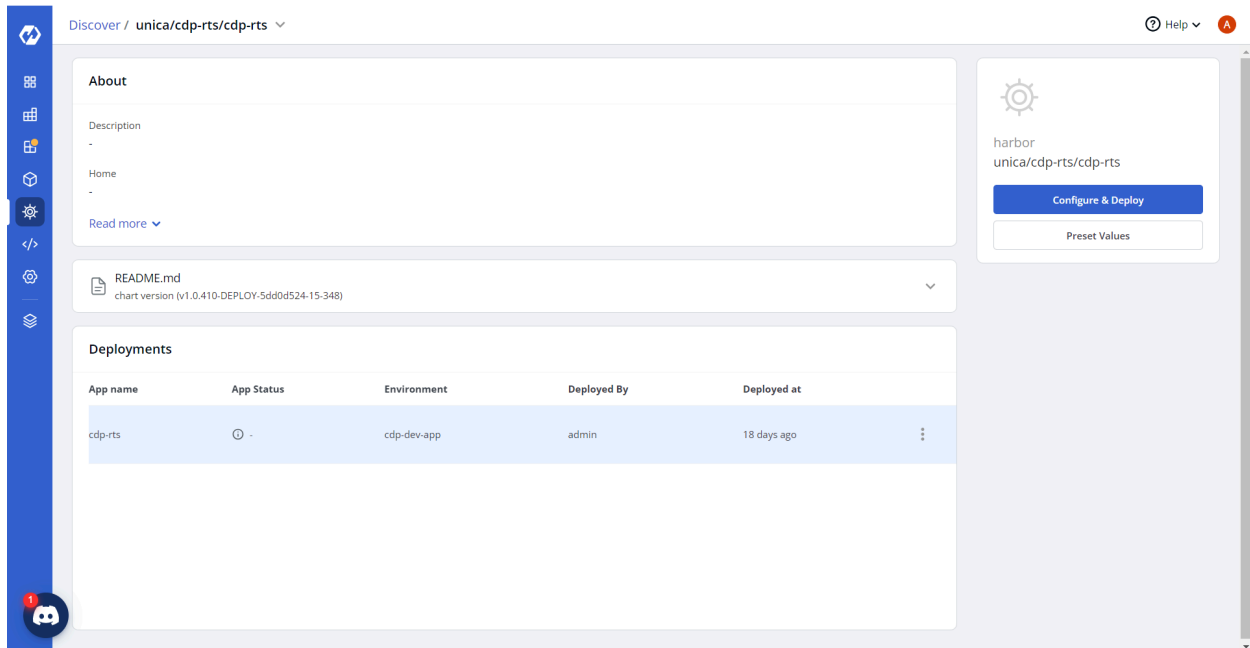
Deploying CDP RTS

To deploy the CDP RTS, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-rtss chart to deploy.



2. Now, configure and deploy the CDP RTS charts.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
FROM_EMAIL: <FROM_EMAIL>
KAFKA_CONSUMER_GROUP_ID: <k8s-rtss-prod>
KAFKA_PRODUCER_FAILURE_TOPIC: <dmp_rt_failure>
KAFKA_PRODUCER_RAMANUJAN_TOPIC: <dmp_rt_ramanujan>
KAFKA_PRODUCER_TOPIC: <dmp_rt_segments>
KAFKA_SST_TOPIC: <dmp_sst_rt_profile>
```

```

KAFKA_TOPIC_LIST: <dmp_sst_rt_profile>
MODEL_CONFIG_LOCAL_PATH: /disk1/rts/config/predictiveSegmentationModel.ini
MODEL_CONFIG_S3_PATH: s3://<s3_bucket_name>/sst/predictivesegments/rtson/predictiveSegmentationModel.ini
MODEL_CUTOFFS_LOCAL_PATH: /disk1/rts/config/predictiveSegmentationCutoffs.ini
MODEL_CUTOFFS_S3_PATH:
  s3://<s3_bucket_name>/sst/predictivesegments/rtson/predictiveSegmentationCutoffs.ini
PMML_MODEL_CONFIG_LOCAL_PATH: /disk1/rts/config/predictivesegments/pmml_model_config/
PMML_MODEL_CONFIG_S3_PATH: s3://<s3_bucket_name>/sst/predictivesegments/pmml_model_config/rtson/
PMML_MODEL_DIRECTORY_LOCAL_PATH: /disk1/rts/config/predictivesegments/pmml_models/
PMML_MODEL_DIRECTORY_S3_PATH: s3://<s3_bucket_name>/sst/predictivesegments/pmml_models/rtson/
TO_EMAIL: <TO_EMAIL>
TSDB_HOST: <TSDB_HOST>
TSDB_PORT: <TSDB_PORT>

```

Discover / unica/cdp-rts/cdp-rts

YAML Manifest output

App Name *

cdp-rts

Project *

cdp-dev-app

Deploy to environment *

cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version

1.0.410-DEPLOY-5dd0d524-15-348

Chart Values

Def... (1.0.410-DEPLOY-5dd0d524-15-348)

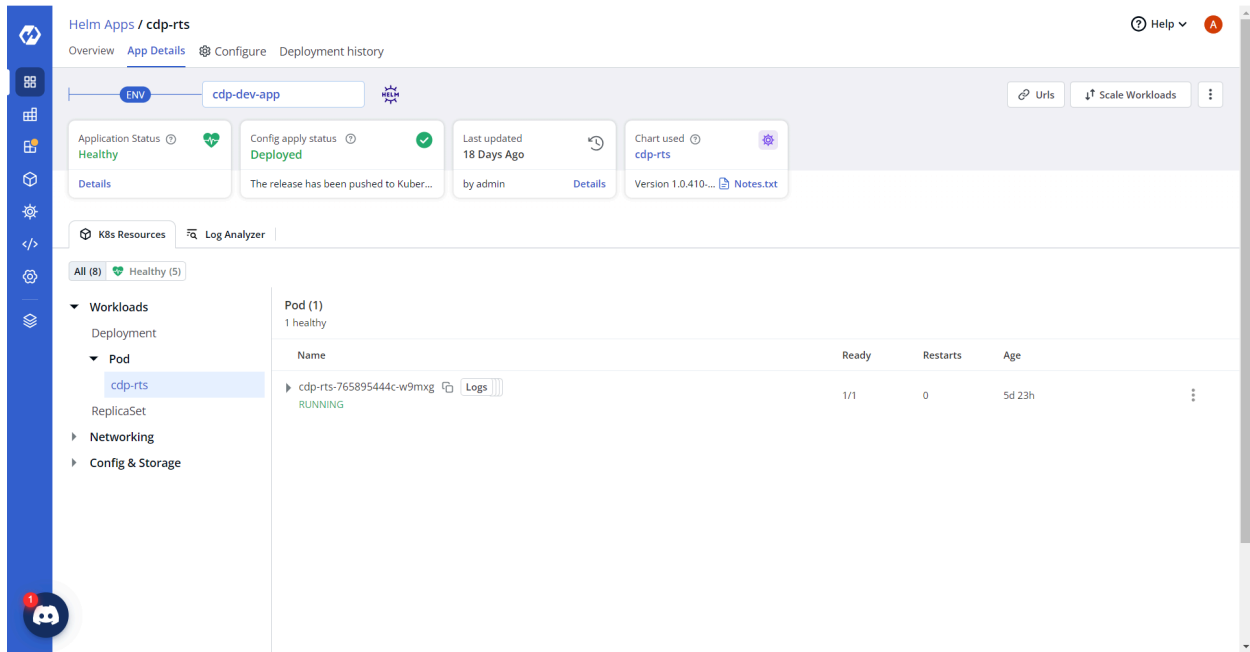
```

1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5         APP_INFRA: k8s
6         APP_REGION: ap-south-1
7         APP_TYPE: realtime
8         AWS_REGION: ap-south-1
9         EXECUTOR_SERVICE_MAX_THREADS: "32"
10        EXECUTOR_SERVICE_MIN_THREADS: "16"
11        EXECUTOR_SERVICE_QUEUE_SIZE: "1000"
12        EXECUTOR_SERVICE_THREAD_TTL: "5"
13        FROM_EMAIL: hcl.com
14        KAFKA_CONSUMER_GROUP_ID: k8s-rts-prod
15        KAFKA_POLL_INTERVAL: "200"
16        KAFKA_PRODUCER_FAILURE_TOPIC: dmp_rt_failure
17        KAFKA_PRODUCER_META: "true"
18        KAFKA_PRODUCER_POOL_SIZE: "16"
19        KAFKA_PRODUCER_RAMANUJAM_TOPIC: dmp_rt_ramanujan
20        KAFKA_PRODUCER_TOPIC: dmp_rt_segments
21        KAFKA_SST_TOPIC: dmp_sst_rt_profile
22        KAFKA_TOPIC_CONSUMER_BATCH_SIZE: "100"
23        KAFKA_TOPIC_CONSUMER_DEFAULT_BATCH_TIMEOUT_MS: "200"
24        KAFKA_TOPIC_CONSUMER_DEFAULT_BATCH_TIMEOUT_TRIGGER: "500"
25        KAFKA_TOPIC_CONSUMERS: "6"
26        KAFKA_TOPIC_LIST: dmp_sst_rt_profile
27        LOG_LEVEL_APP: error
28        LOG_LEVEL_METRICS: "off"
29        LOG_LEVEL_ROOT: error
30        LOG_LEVEL_UTILS: "off"
31        MAX_BATCH_EXECUTE_RETRIES: "3"
32        MAX_EVENT_BATCH_LAG: "64"
33        MODEL_CONFIG_LOCAL_PATH: /disk1/rts/config/predictiveSegmentationModel.ini
34        MODEL_CONFIG_S3_PATH: s3://hclsw-hxcd-hss-ned-s3-rdn-ann/sst/predictivesegments/rtson/predictiveSegmentationModel.ini

```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

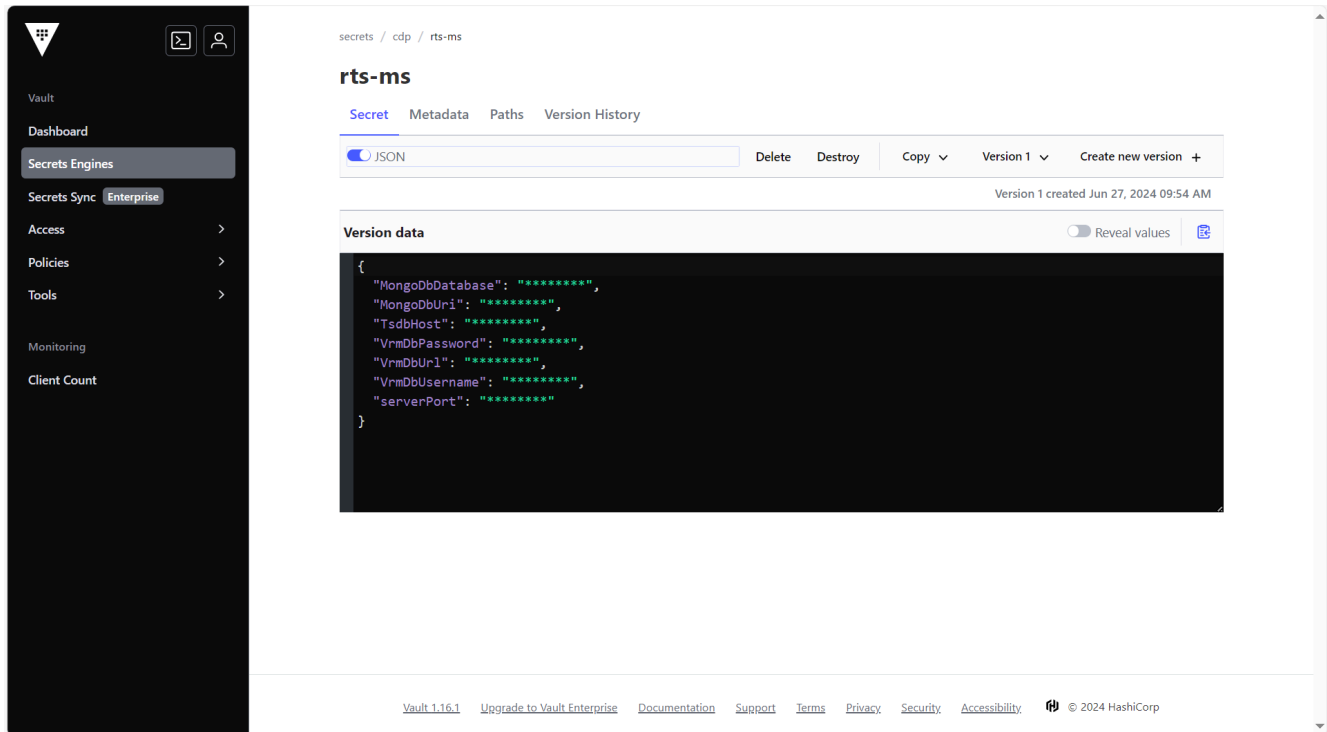


Deploying CDP RTS MD

This section provides detailed instructions on how to deploy HCL CDP RTS MS using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP RTS MS.



To create the UI secret in HashiCorp Vault, follow the steps below:

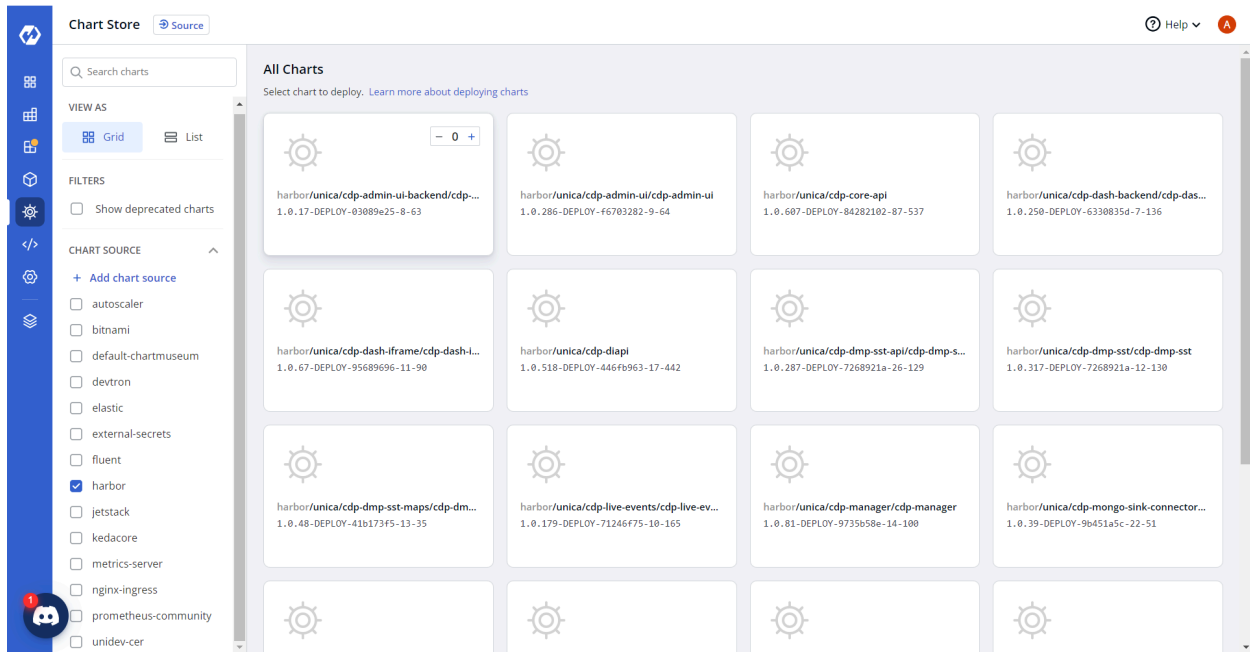
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "MongoDbDatabase": "<MongoDbDatabase>",
  "MongoDbUri": "mongodb://<user>:<password>@<ip>:<port>/?directConnection=true",
  "TsdbHost": "<TsdbHost>",
  "VrmDbPassword": "<TsdbHost>",
  "VrmDbUrl": "jdbc:mariadb://<ip>:<port>/<dbName>?autoReconnect=true",
  "VrmDbUsername": "<VrmDbUsername>",
  "serverPort": "<serverPort>"
}
```

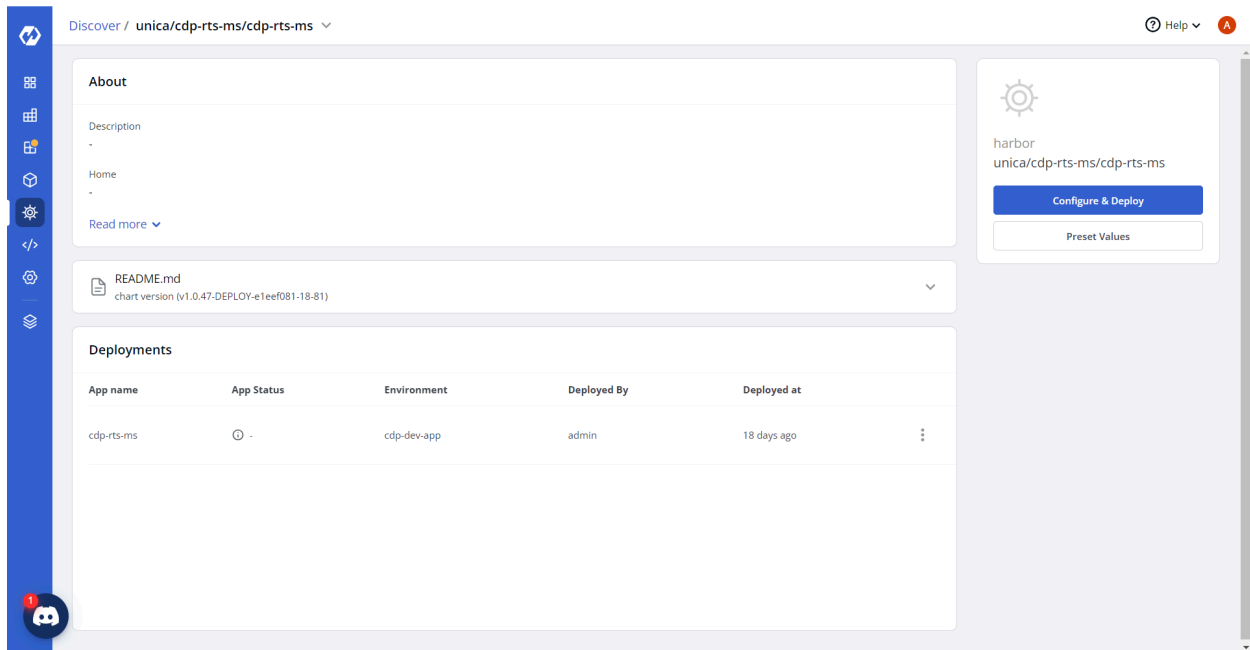
Deploying CDP RTS

To deploy the CDP RTS, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-rtms chart to deploy.

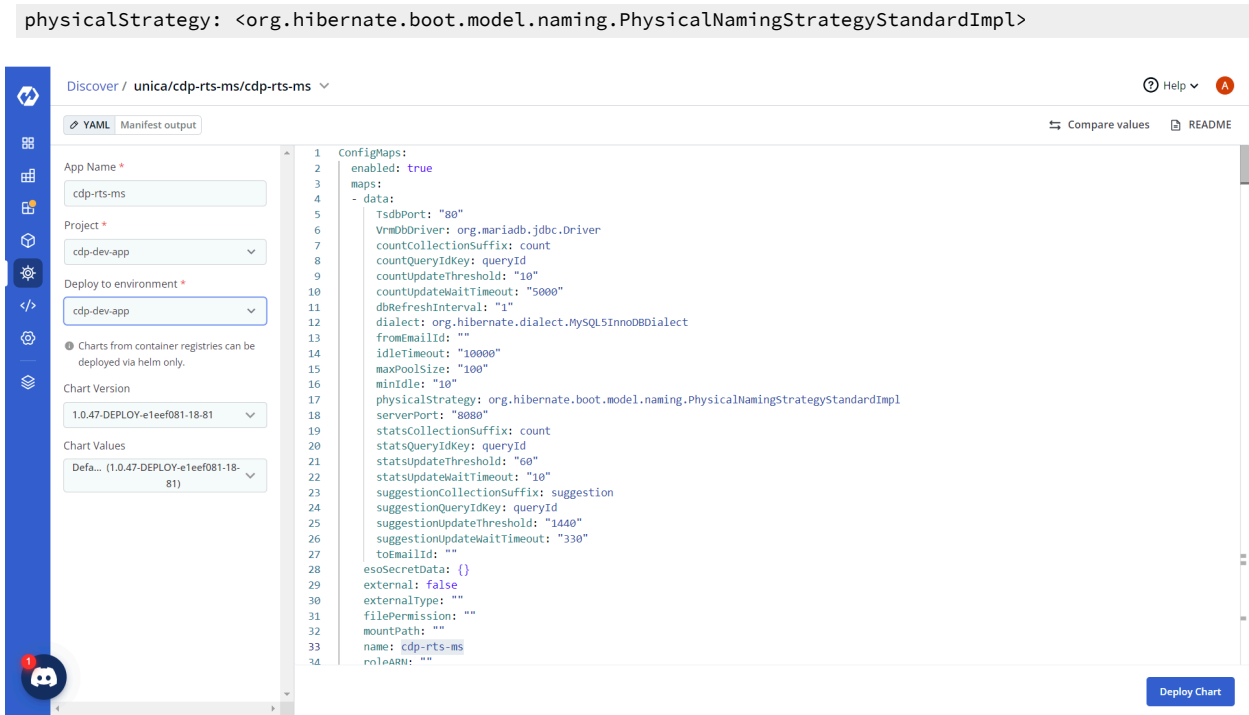


2. Now, configure and deploy the CDP RTS charts.

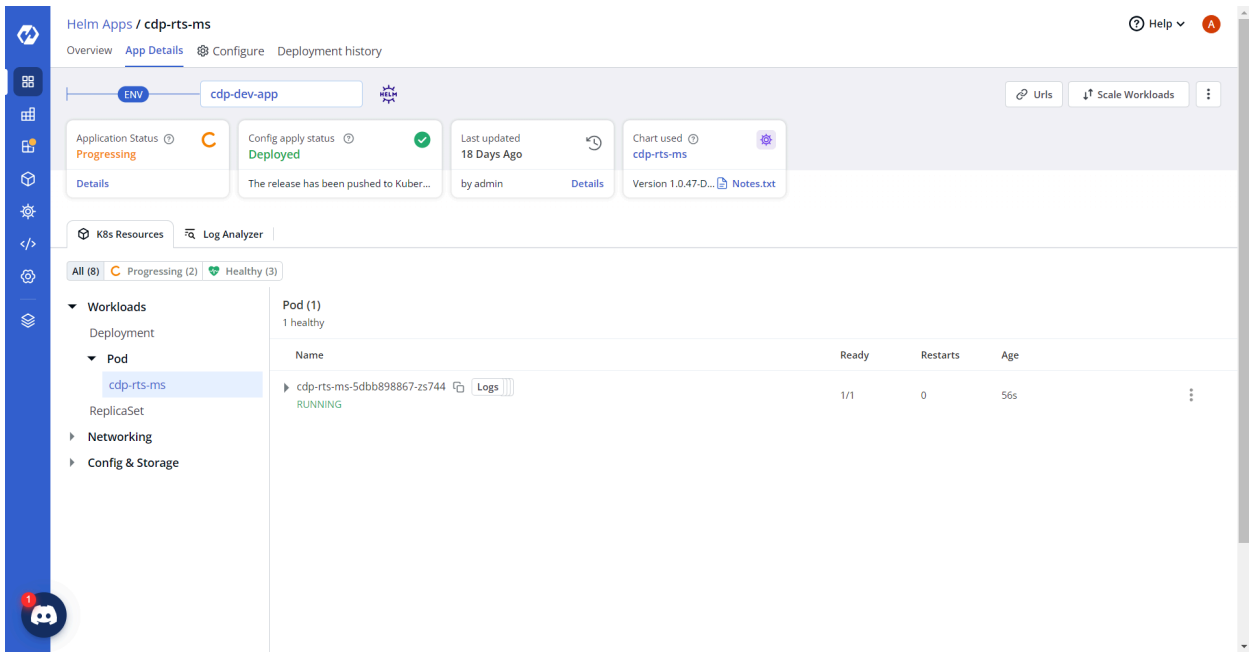


3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
FROM_EMAIL: <FROM_EMAIL>
TO_EMAIL: <TO_EMAIL>
TSDB_HOST: <TSDB_HOST>
TSDB_PORT: <TSDB_PORT>
VrmDbDriver: <org.mariadb.jdbc.Driver>
dialect: <org.hibernate.dialect.MySQL5InnoDBDialect>
```



4. On successful deployment, validate the deployment as shown below.

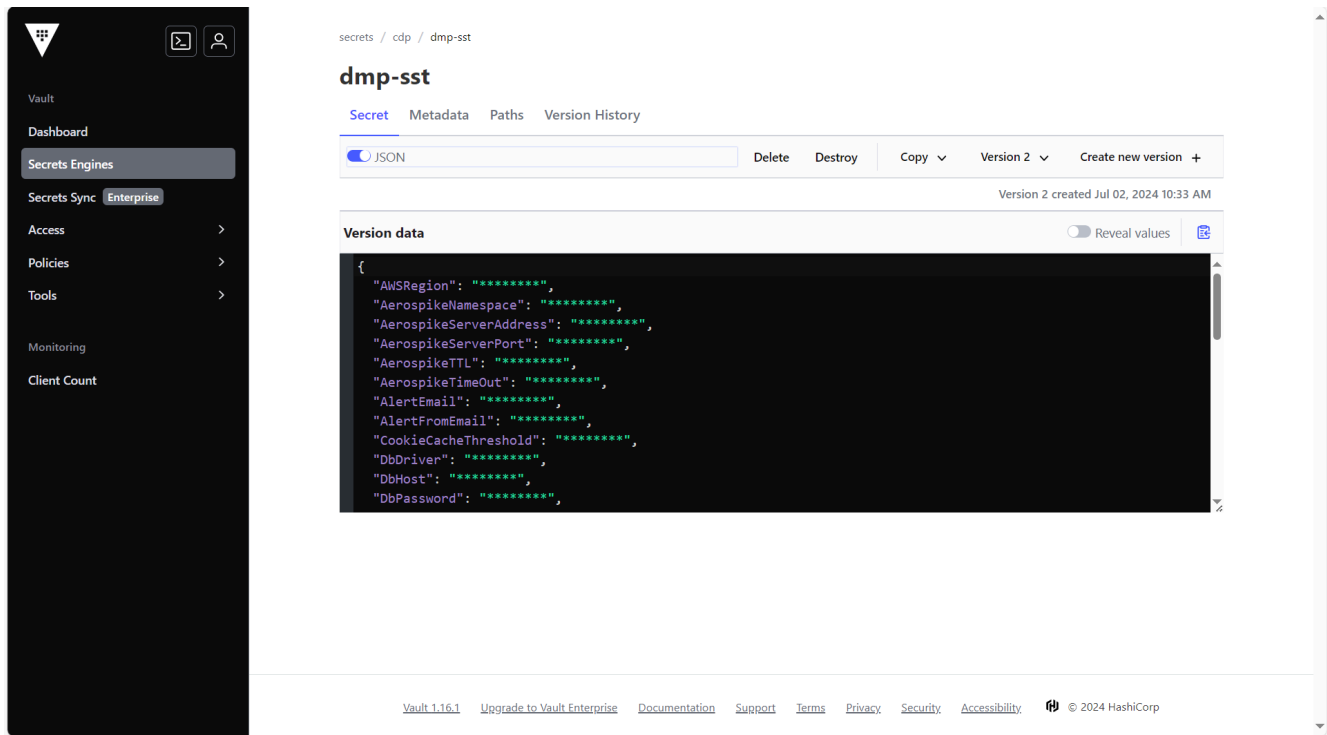


Deploying CDP DMP SST

This section provides detailed instructions on how to deploy HCL CDP DMP SST using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP DMP SST.



To create the UI secret in HashiCorp Vault, follow the steps below:

1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "AWSRegion": "<AWSRegion>",
  "AerospikeNamespace": "<AerospikeNamespace>",
  "AerospikeServerAddress": "<AerospikeServerAddress>",
  "AerospikeServerPort": "<AerospikeServerPort>",
  "AerospikeTTL": "15552000",
  "AerospikeTimeOut": "10000",
  "AlertEmail": "",
  "AlertFromEmail": "",
  "CookieCacheThreshold": "60",
  "DbDriver": "<DbDriver>",
  "DbHost": "<DbHost>",
  "DbPassword": "<DbPassword>",
  "DbPort": "<DbPort>",
  "DbUrl": "jdbc:mariadb://ip:port/dbName",
  "DbUsername": "<DbUsername>",
  "GraphDBEnable": "false",
  "GraphDBEndpoint": "<GraphDBEndpoint>",
  "GraphDBPort": "<GraphDBPort>",
  "HBaseDBEnable": "false",
  "HbaseZookeeperClientPort": "",
  "HbaseZookeeperQuorum": "",
  "KafkaBootstrapServer": "<KafkaBootstrapServer>",
```

```

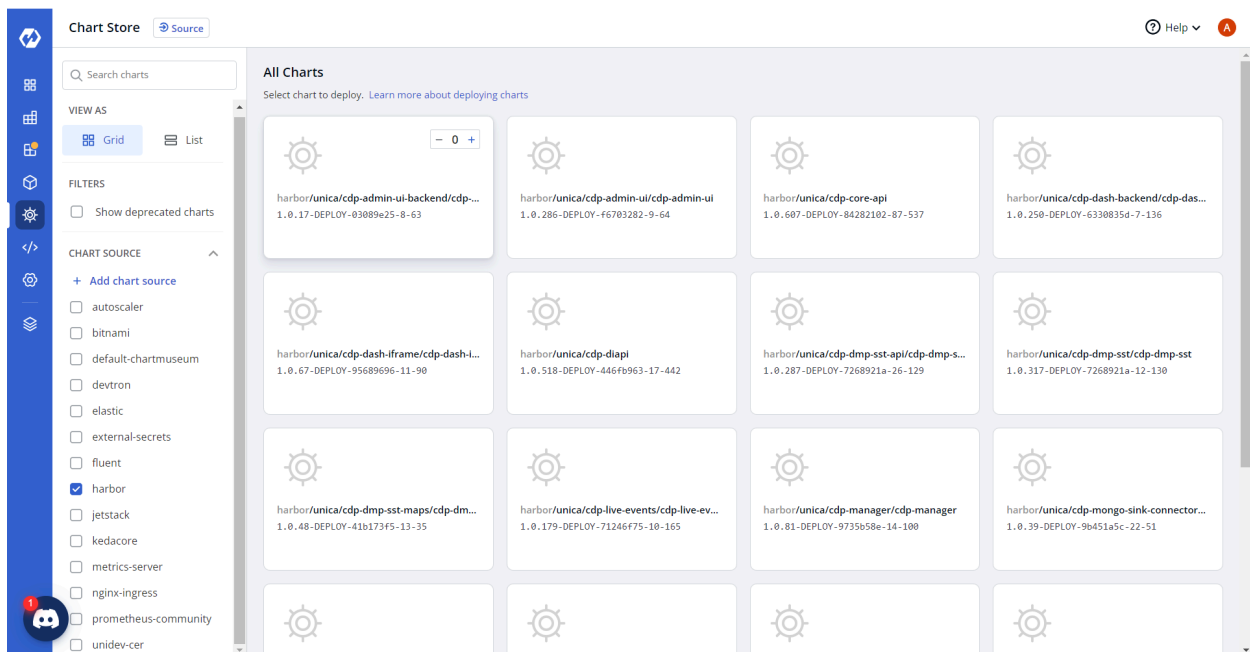
"KafkaConsumerGroup": "<dmp-sst-kafka-k8>",
"KafkaPassword": "<KafkaPassword>",
"KafkaProducerHostName": "<KafkaPassword>",
"KafkaProducerTopic": "<dmp_sst_rt_profile>",
"KafkaProfileProducerTopic": "<dmp_sst_profile>",
"KafkaTopicConsumerBatchSize": "<10,10,10,10>",
"KafkaTopicConsumerDefaultBatchTimeoutMS": "<5000,5000,5000,5000>",
"KafkaTopicConsumerDefaultBatchTimeoutTrigger": "<60000,10000,10000,10000>",
"KafkaTopicConsumers": "<6,6,6,6>",
"KafkaTopicList": "<dmp_sst_nba,dmp_sst_nba_di_api,analyze_post,dmp_sst_data>",
"KafkaUsername": "<KafkaUsername>",
"SstApiAuthKeys": "<HdxDeMMHvUyrqlSBHyaCaQGA>",
"TsdbHost": "<TsdbHost>",
"TsdbPort": "<TsdbPort>",
"ZookeeperClients": "<15>",
"ZookeeperLockAddress": "<cdp-dmp-sst-zookeeper.cdp-dev-app:2181>",
"ZookeeperLockAddressOld": "",
"ZookeeperRetrySleepTime": "<1000>",
"region": "<region>",
"type": "<realtime>"
}

```

Deploying CDP DMP SST

To deploy the CDP DMP SST, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-dmp-sst chart to deploy.



2. Now, configure and deploy the CDP DMP SST charts.

Discover / unica/cdp-dmp-sst/cdp-dmp-sst

About

Description
-

Home
-

[Read more](#)

README.md
chart version (v1.0.317-DEPLOY-7268921a-12-130)

Deployments

App name	App Status	Environment	Deployed By	Deployed at
cdp-dmp-sst	○ -	cdp-dev-app	admin	7 days ago

Configure & Deploy

Preset Values

3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
ValidValuesConfig: /disk1/config/dmp-sst-maps/valid_values.json
ValidValuesS3Path: s3://<bucket_name>/dmp/config/predictivesegments/validvalues.tsv
ValuesMappingConfig: /disk1/config/dmp-sst-maps/values_mapping.json
ValuesMappingS3Path: s3://<bucket_name>/dmp/config/predictivesegments/mappingvalues.tsv
```

Discover / unica/cdp-dmp-sst/cdp-dmp-sst

YAML Manifest output

App Name *

cdp-dmp-sst

Project *

cdp-dev-app

Deploy to environment *

cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version

1.0.317-DEPLOY-7268921a-12-130

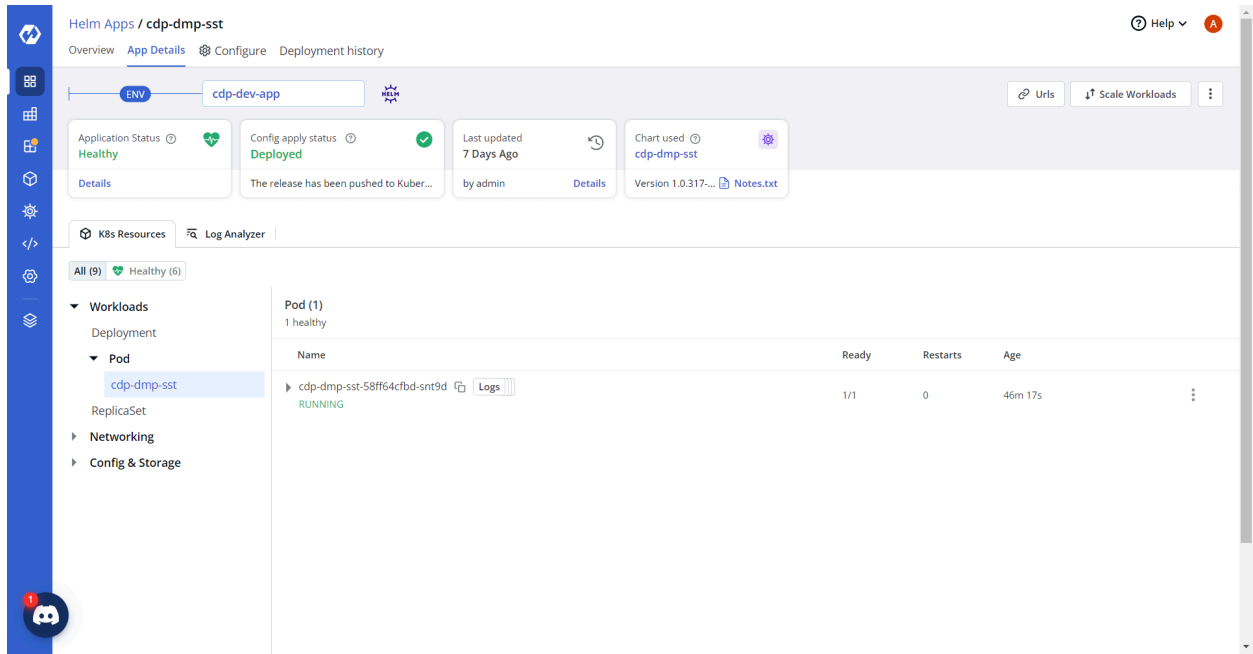
Chart Values

Def... (1.0.317-DEPLOY-7268921a-12-130)

```
1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5       CampaignIncludeList: VIZVRM6499\,VIZVRM6529\,VIZVRM6525
6       DBDataOidsCapLimit: "25"
7       Dbname: vrm
8       DemoCampaignList: VIZVRM6515\,VIZVRM6489
9       EncryptionEnabled: "true"
10      ExecutorServiceMaxThreads: "256"
11      ExecutorServiceMinThreads: "64"
12      ExecutorServiceQueueSize: "10000000"
13      ExecutorServiceThreadTTL: "5"
14      GeoISPFFile: /disk1/dmp-sst-maps/GeoIPISP.dat
15      GeoLiteCityFile: /disk1/dmp-sst-maps/GeoLite2-City.mmdb
16      GmondCollectors: ""
17      GmondPort: "8652"
18      HostNameOverride: dmpsst
19      KafkaAnalyzePostTopic: analyze_post
20      KafkaAppPushCleanupTopic: apppush_cleanup
21      KafkaDmpRtSegmentsRefreshTopic: dmp_rt_segments_refresh
22      KafkaDmpRtSegmentsTopic: dmp_rt_segments_offline
23      KafkaDmpSegmentsTopic: dmp-segments
24      KafkaDmpSstBidTopic: dmp_sst_bid
25      KafkaDmpSstDataTopic: dmp_sst_data
26      KafkaDmpSstImprTopic: dmp_sst_impr
27      KafkaDmpSstMetricTopic: dmp_sst_metrics
28      KafkaDmpSstNbaDiApiTopic: dmp_sst_nba_di_api
29      KafkaDmpSstNbaTopic: dmp_sst_nba
30      KafkaDmpSstProfileTopic: dmp_sst_profile
31      KafkaDmpSstProgClicksTopic: dmp_sst_prog_click
32      KafkaDmpSstProgImprTopic: dmp_sst_prog_impr
33      KafkaDmpSstRtProfileTopic: dmp_sst_rt_profile
34      KafkaDmpSstRtProfileTopic: onsite_notif_click
```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

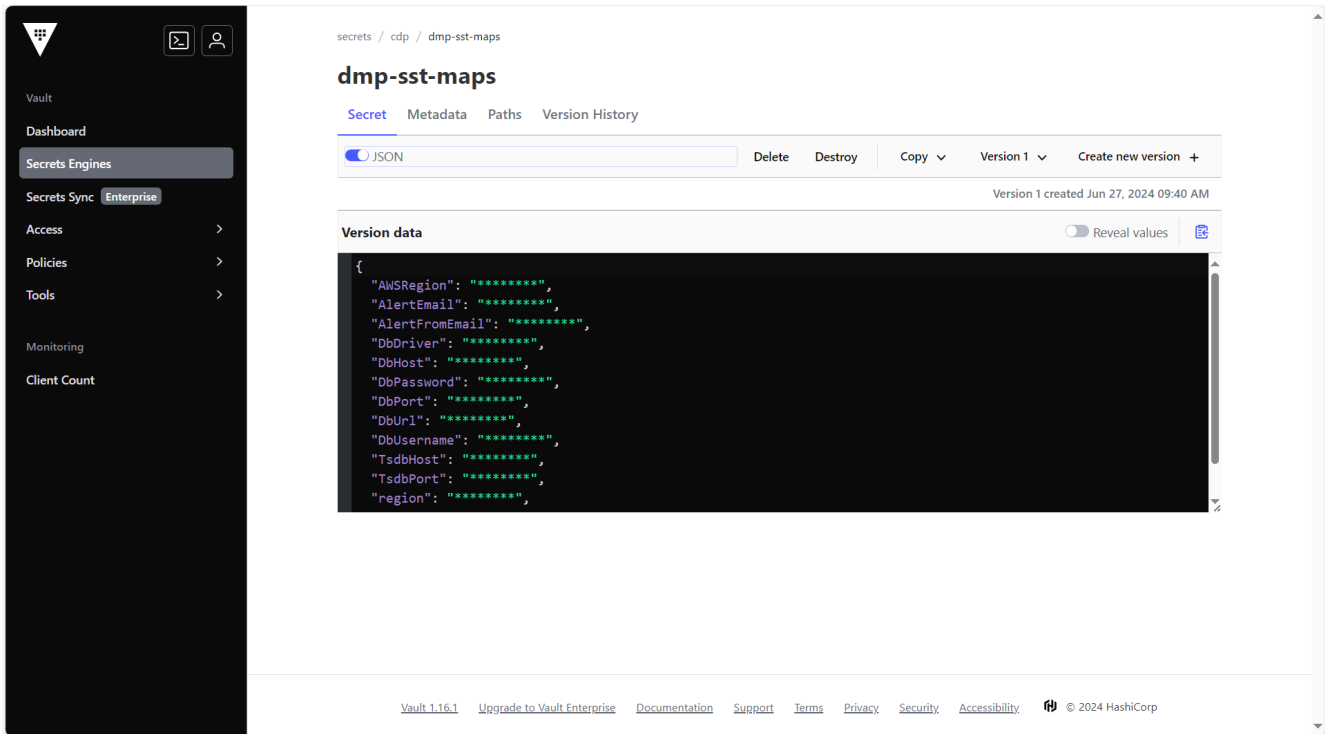


Deploying CDP DMP SST Maps

This section provides detailed instructions on how to deploy HCL CDP DMP SST Maps using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP DMP SST Maps.



To create the UI secret in HashiCorp Vault, follow the steps below:

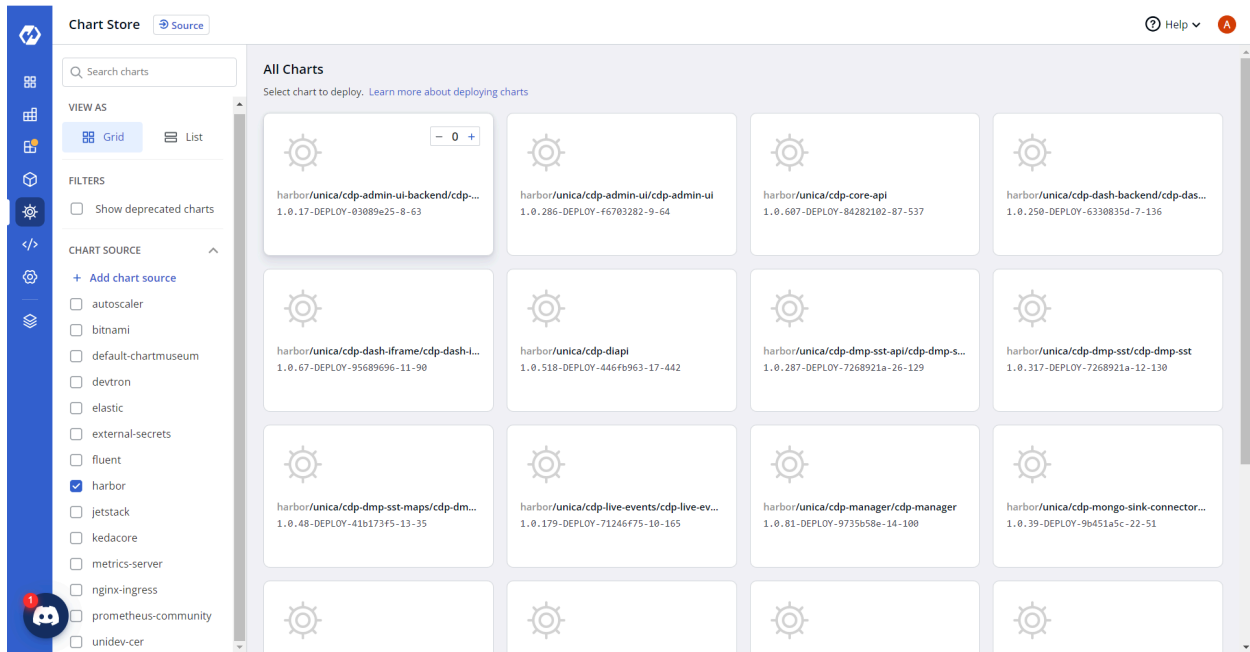
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "AWSRegion": "<AWSRegion>",
  "AlertEmail": "",
  "AlertFromEmail": "",
  "DbDriver": "<com.mysql.cj.jdbc.Driver>",
  "DbHost": "<DbHost>",
  "DbPassword": "<DbPassword>",
  "DbPort": "<DbPort>",
  "DbUrl": "jdbc:mysql://ip:port/dbName",
  "DbUsername": "<DbUsername>",
  "TsdbHost": "<TsdbHost>",
  "TsdbPort": "<TsdbPort>",
  "region": "<region>",
  "type": "<b-aero-7-nodes>"
}
```

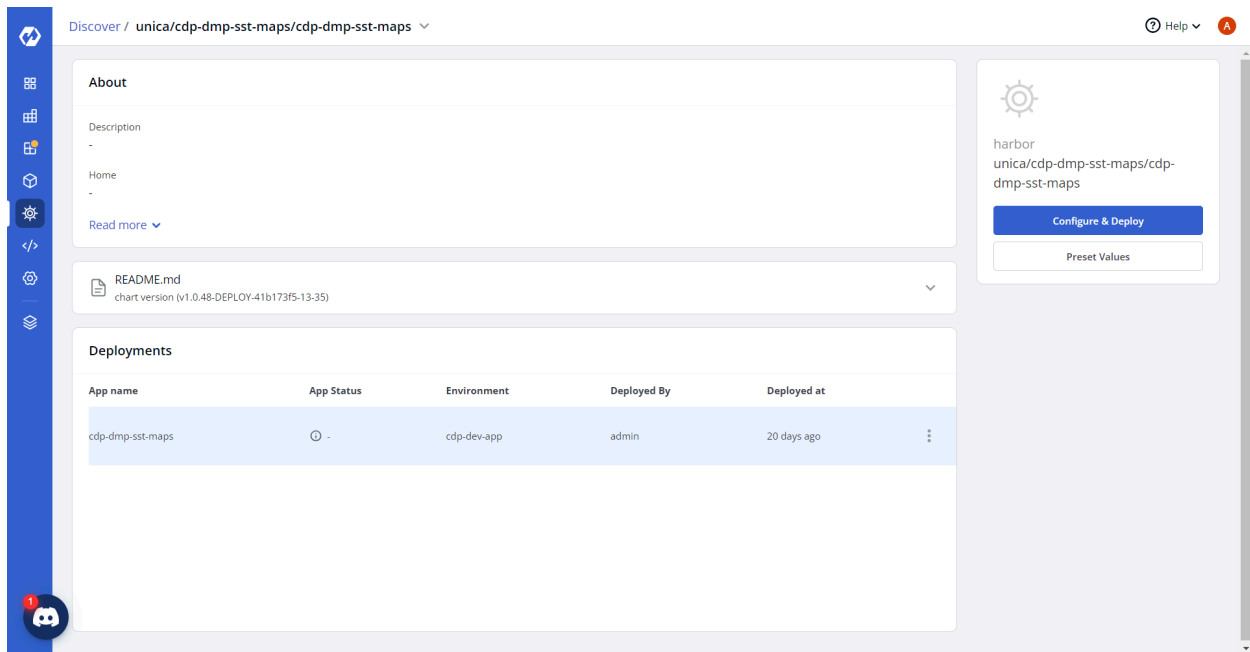
Deploying CDP DMP Maps

To deploy the CDP DMP Maps, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-dmp-sst-maps chart to deploy.

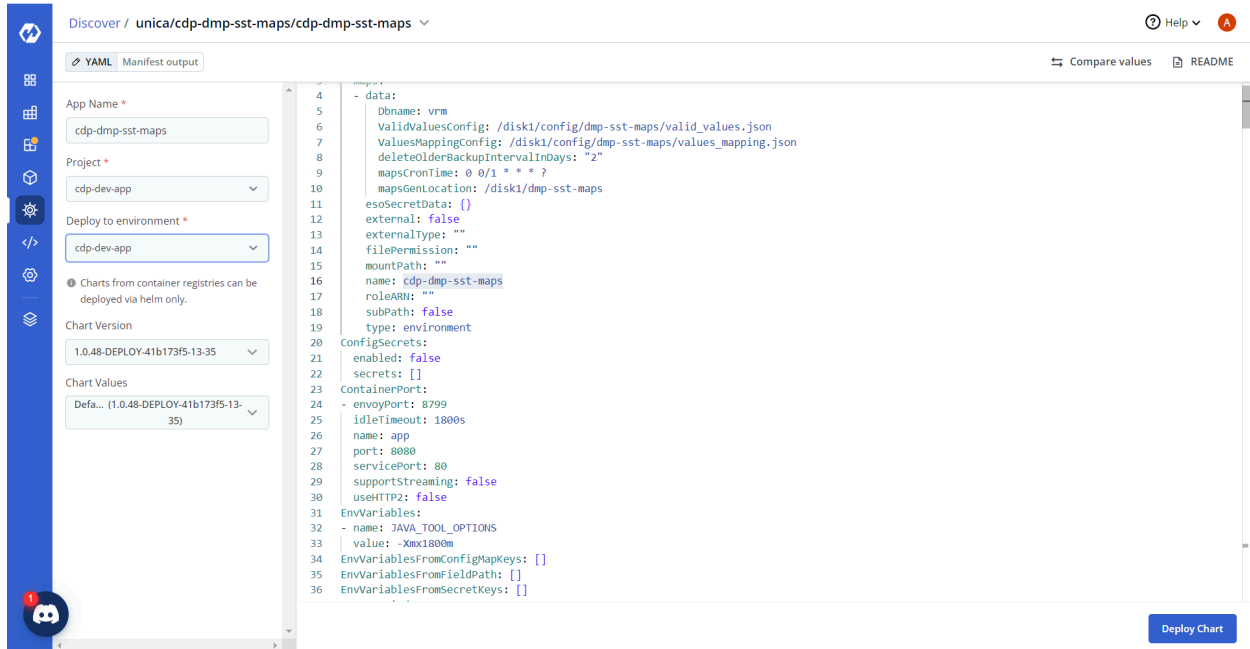


2. Now, configure and deploy the CDP DMP SST Maps charts.

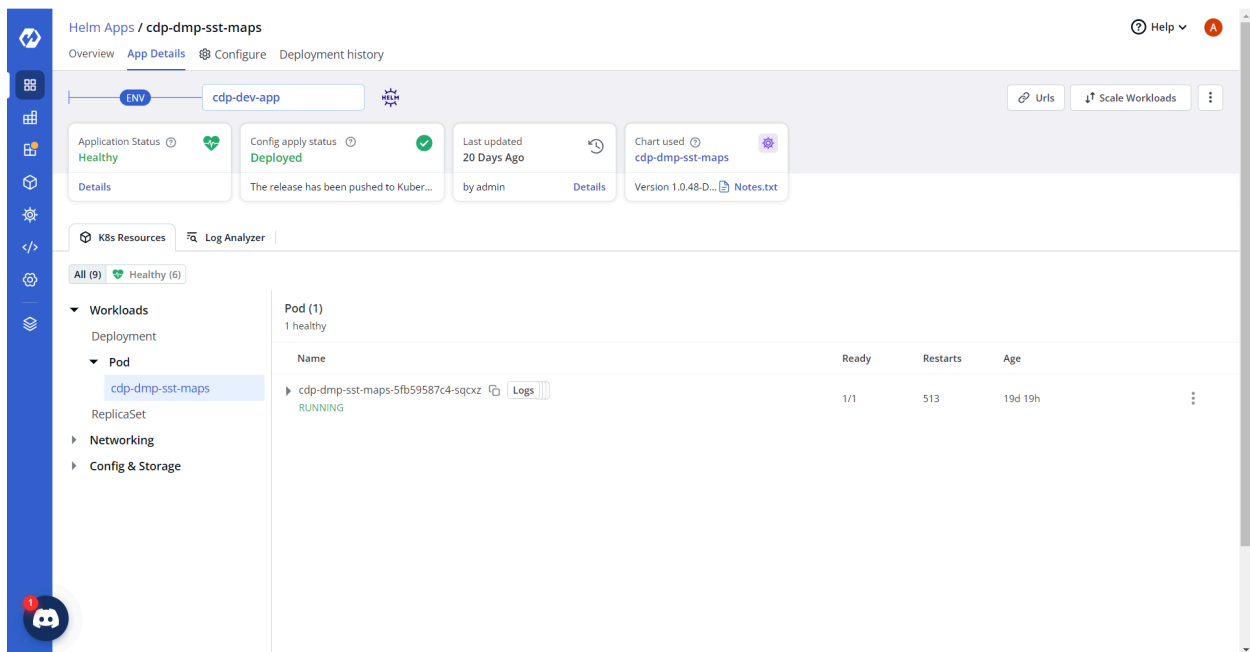


3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
ValidValuesConfig: /disk1/config/dmp-sst-maps/valid_values.json
ValuesMappingConfig: /disk1/config/dmp-sst-maps/values_mapping.json
mapsGenLocation: /disk1/dmp-sst-maps
DbName: <DbName>
```



4. On successful deployment, validate the deployment as shown below.

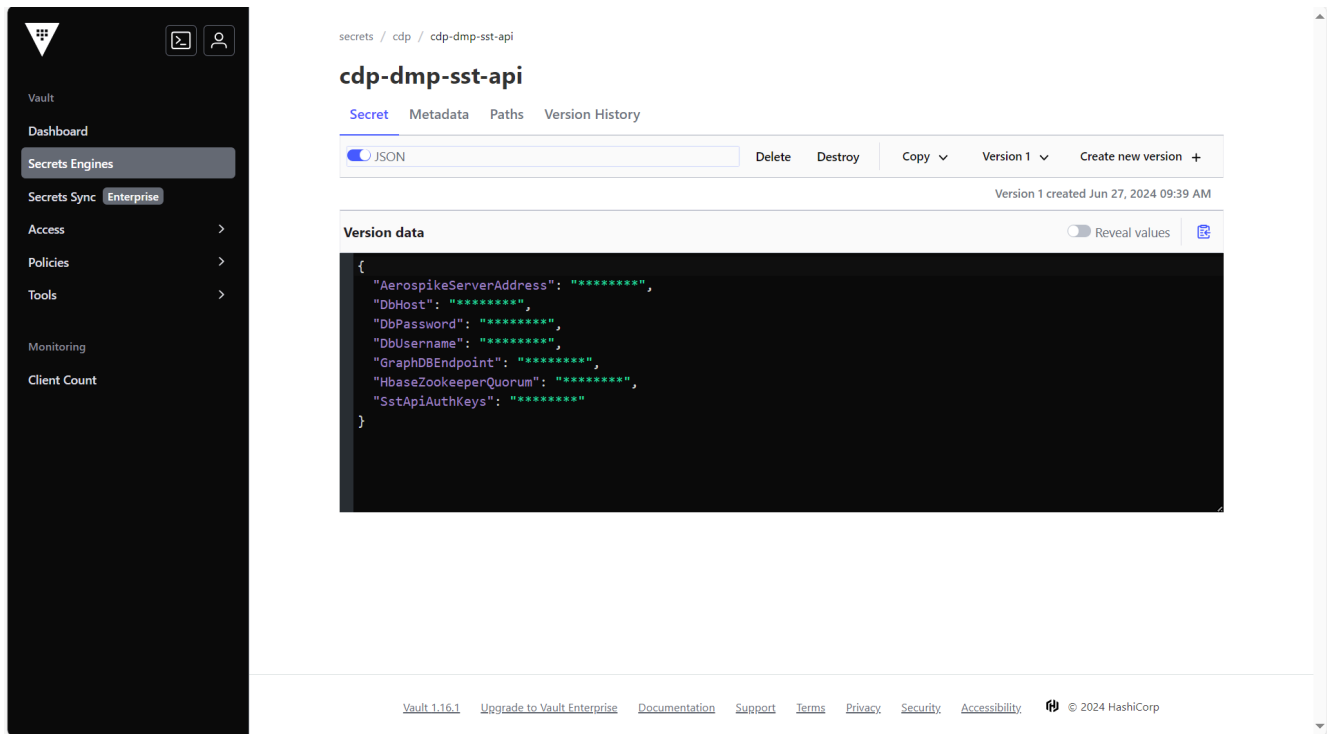


Deploying CDP DMP SST API

This section provides detailed instructions on how to deploy HCL CDP DMP SST API using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP DMP SST API.



To create the UI secret in HashiCorp Vault, follow the steps below:

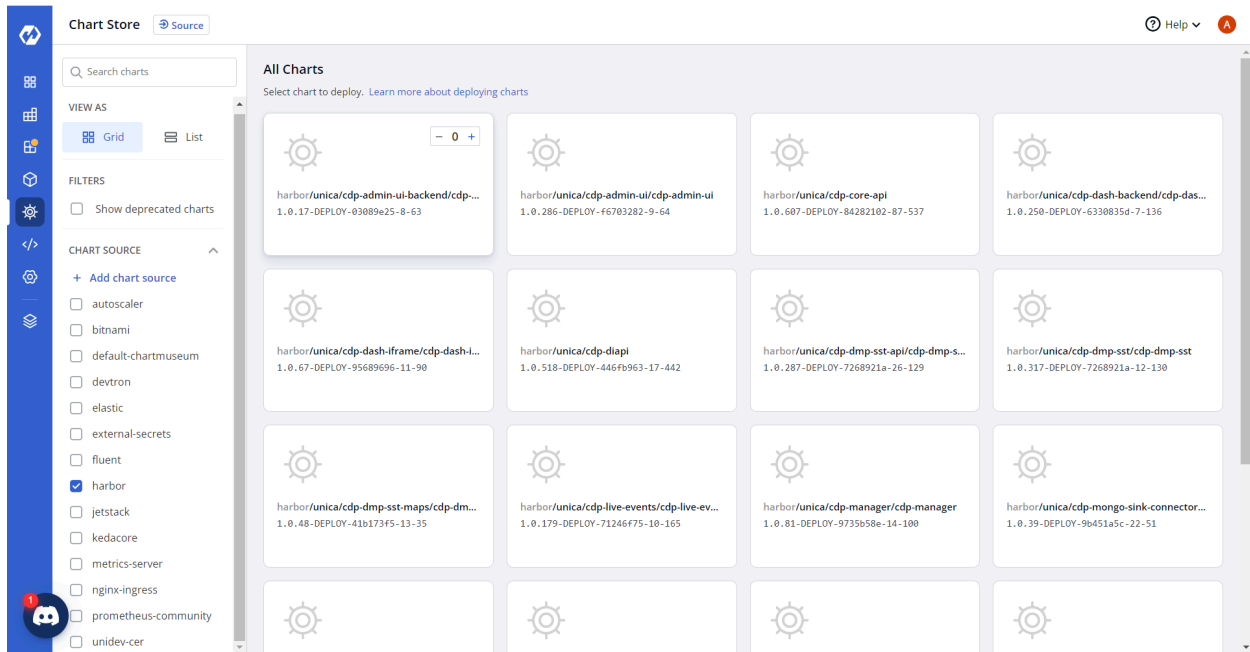
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "AerospikeServerAddress": "<AerospikeServerAddress>",
  "DbHost": "<DbHost>",
  "DbPassword": "<DbPassword>",
  "DbUsername": "<DbUsername>",
  "GraphDBEndpoint": "",
  "HbaseZookeeperQuorum": "",
  "SstApiAuthKeys": "<SstApiAuthKeys>"
}
```

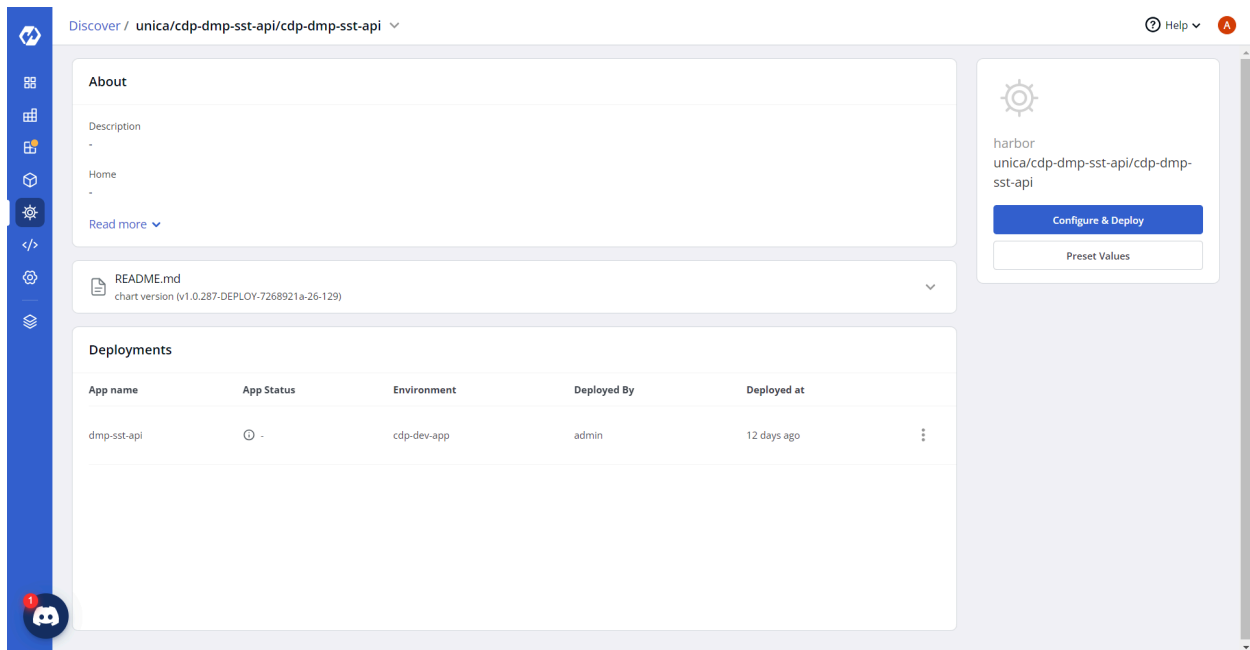
Deploying CDP DMP SST API

To deploy the CDP DMP SST API, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the `cdp-dmp-sst-api` chart to deploy.

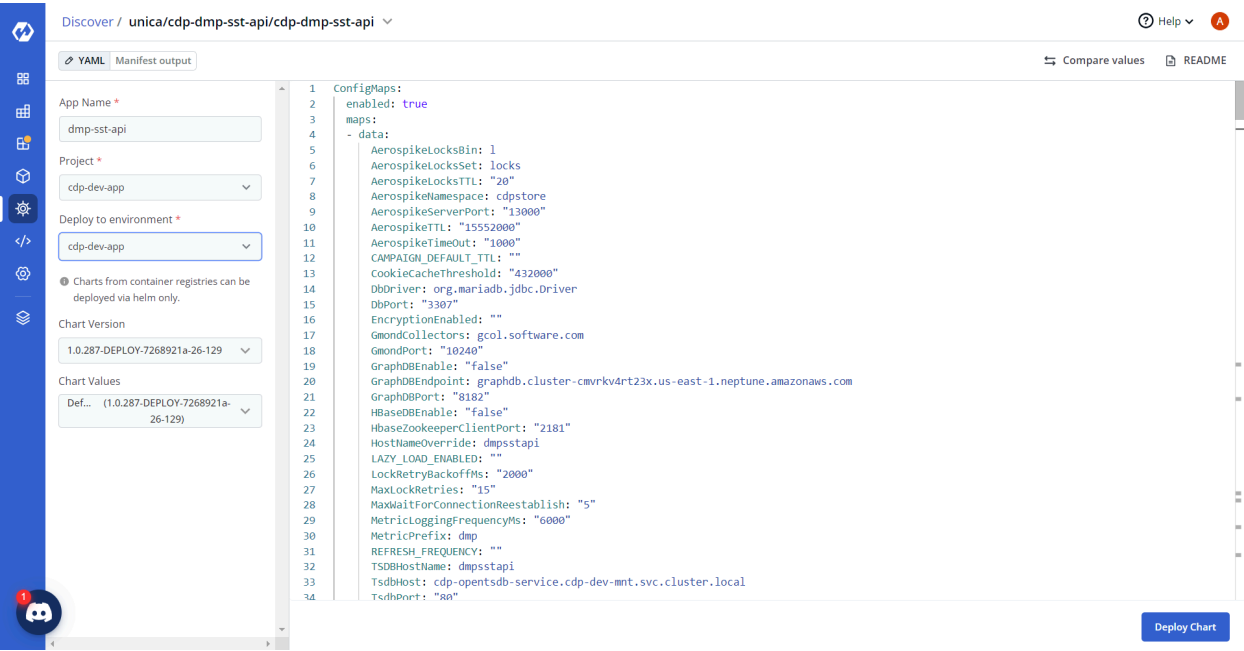


2. Now, configure and deploy the CDP DMP SST API charts.

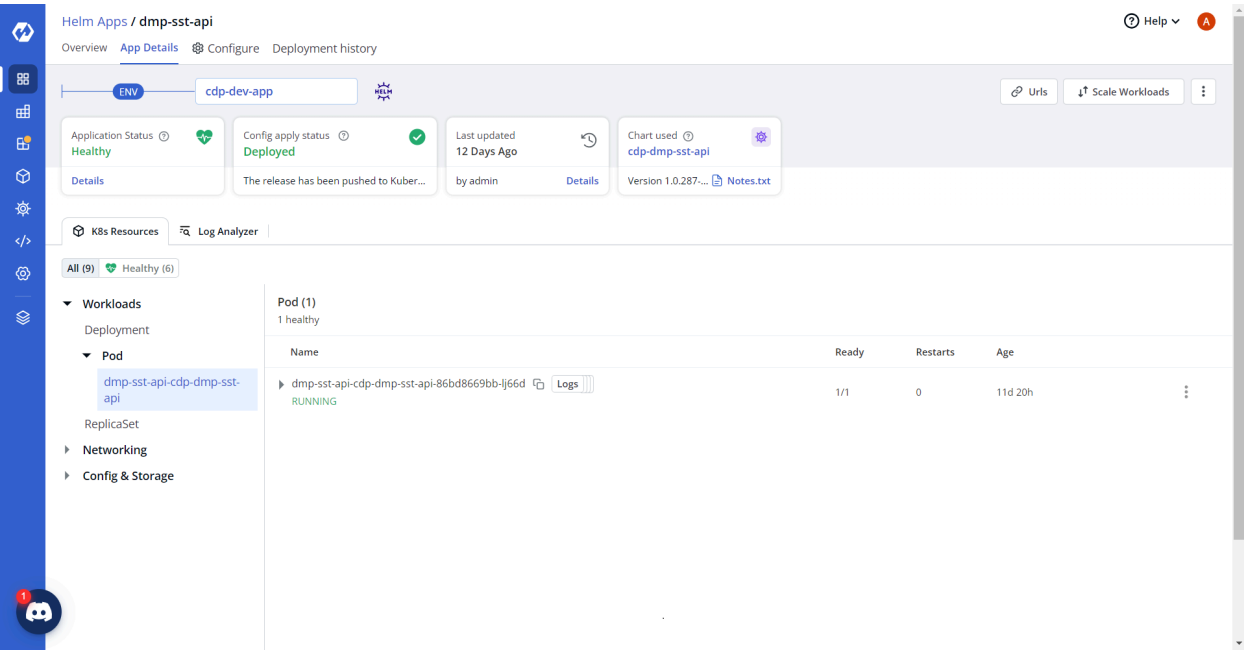


3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
DbDriver: <DbDriver>
DbPort: <DbPort>
EncryptionEnabled: ""
TsdbHost: <TsdbHost>
TsdbPort: <TsdbPort>
```



4. On successful deployment, validate the deployment as shown below.



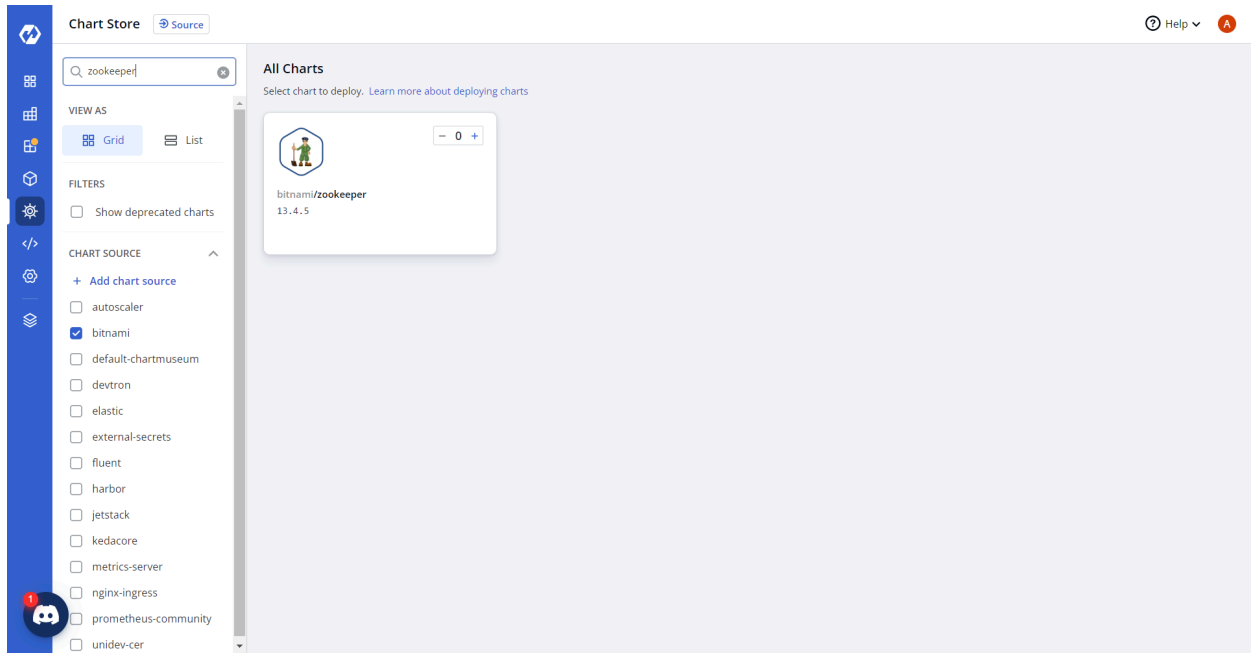
Deploying CDP DMP SST Zookeeper

This section provides detailed instructions on how to deploy HCL CDP DMP SST Zookeeper using the Devtron in the OpenShift.

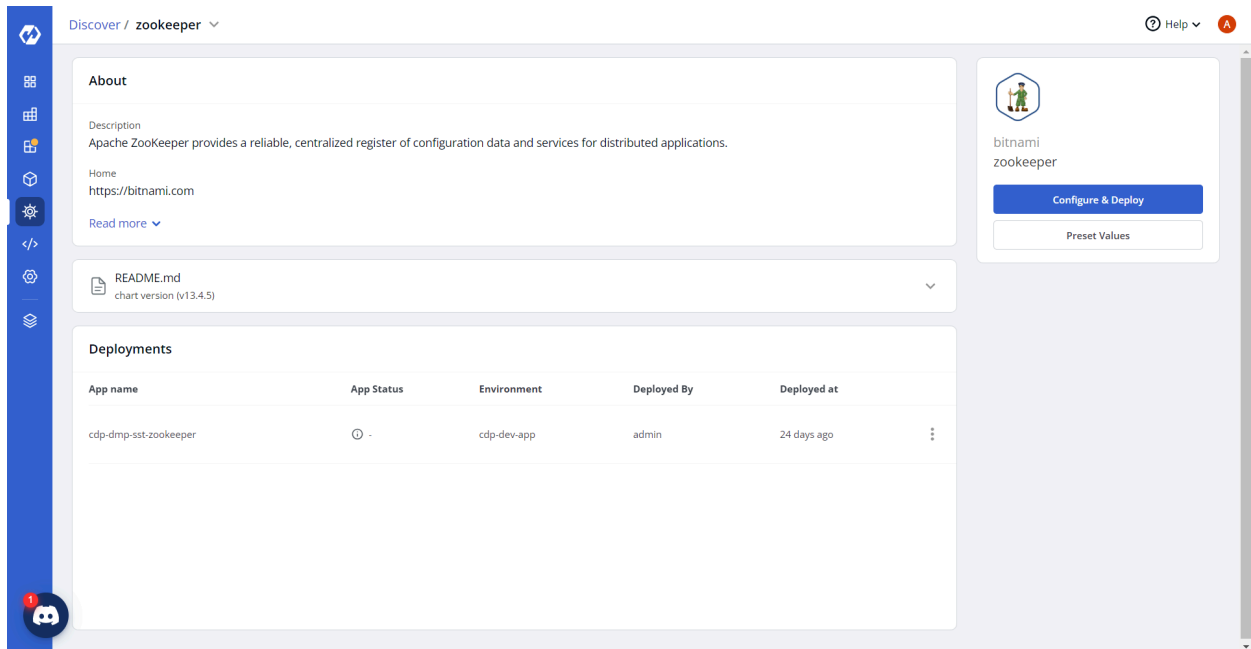
Deploying CDP DMP SST Zookeeper

To deploy the CDP DMP SST Zookeeper, follow these steps below:

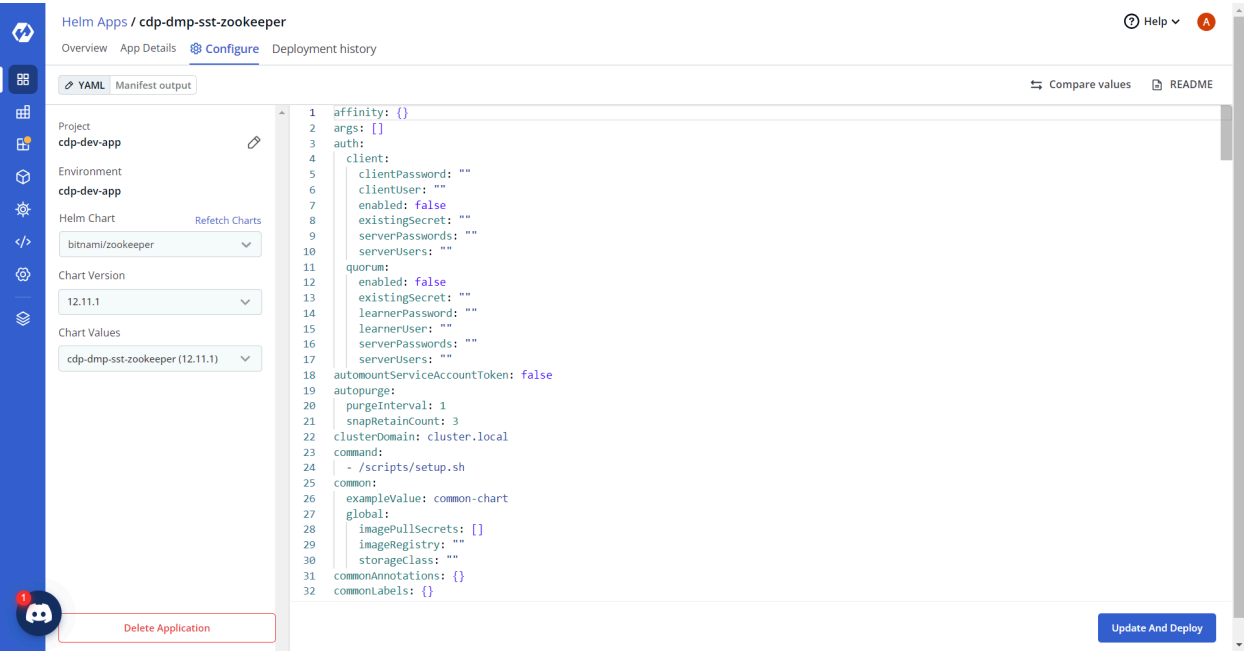
1. Navigate to the Devtron **Chart Store**, and select the bitnami/zookeeper chart to deploy.



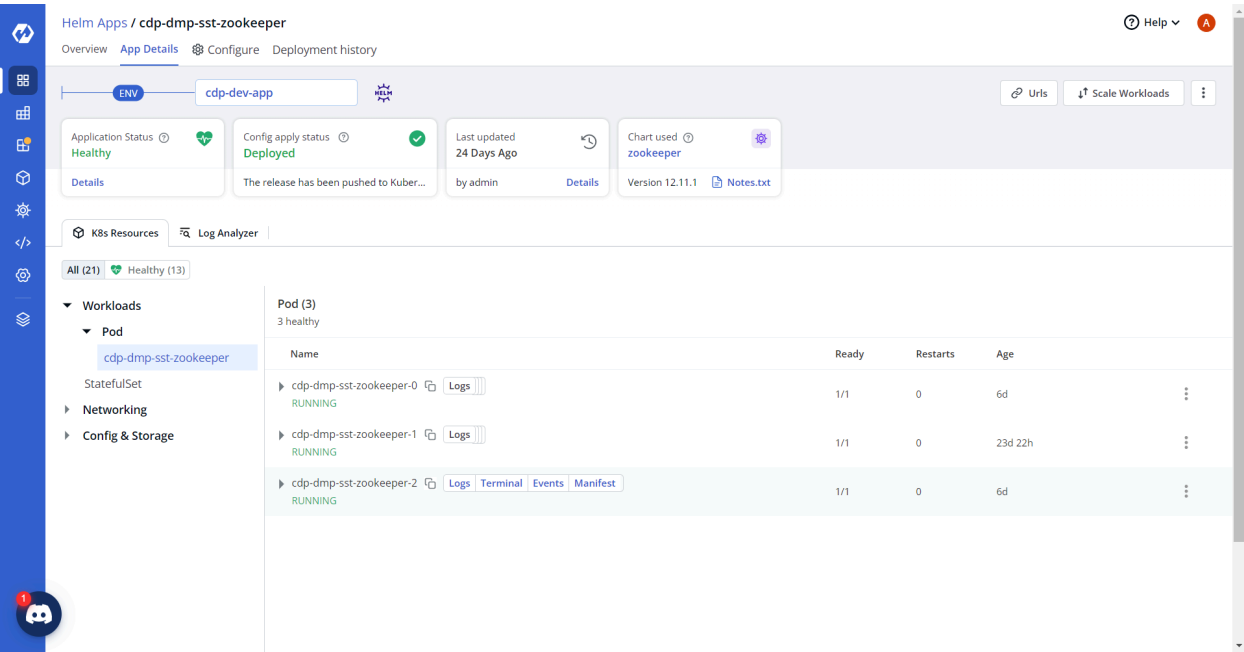
2. Now, configure and deploy the CDP DMP SST Zookeeper charts.



3. In the **YAML** section, update the configuration, and deploy the charts.



4. On successful deployment, validate the deployment as shown below.

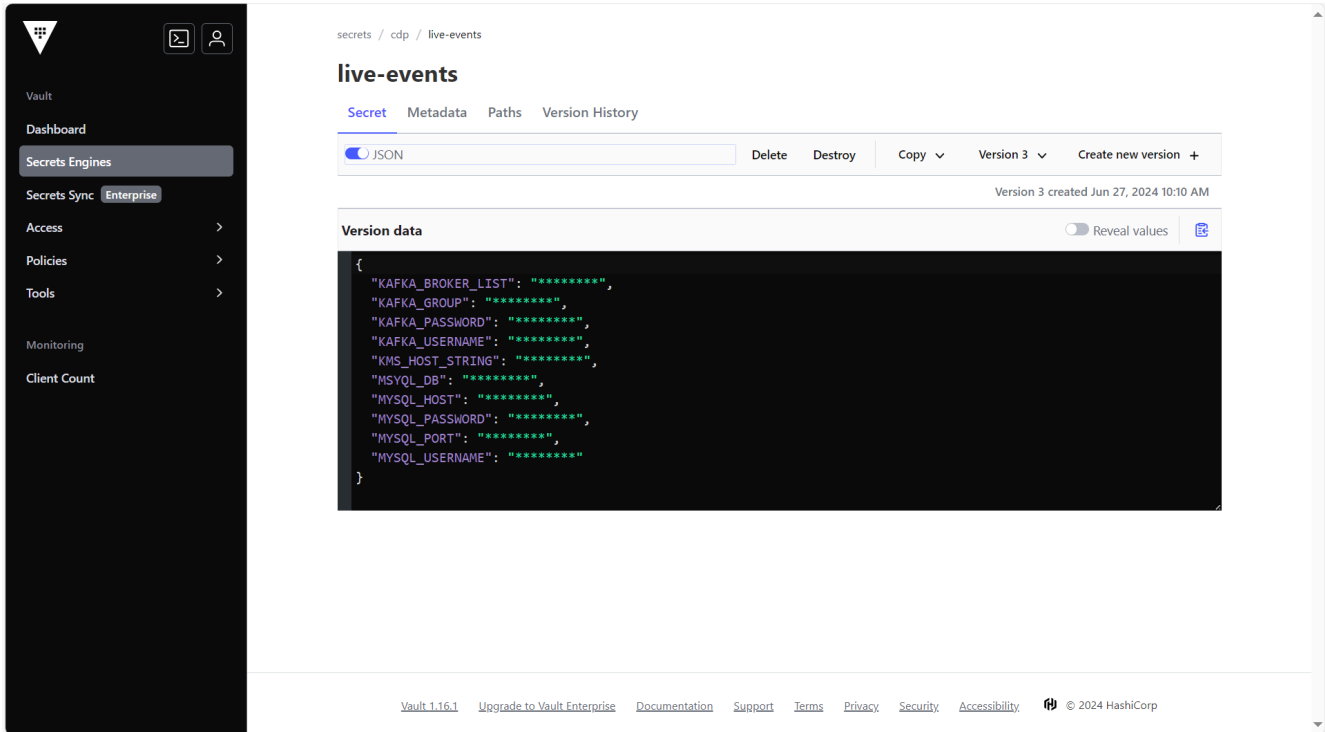


Deploying CDP Live Events

This section provides detailed instructions on how to deploy HCL CDP Live Events using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Live Events.



To create the UI secret in HashiCorp Vault, follow the steps below:

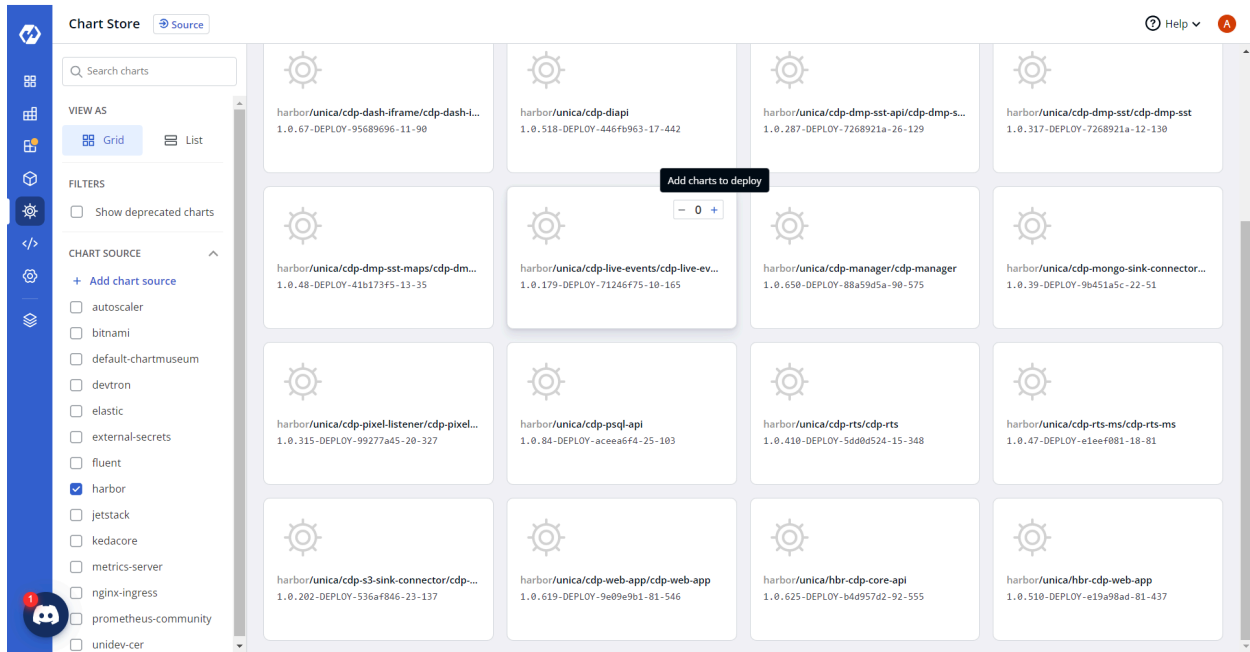
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "KAFKA_BROKER_LIST": "<KAFKA_BROKER_LIST>",
  "KAFKA_GROUP": "<KAFKA_GROUP>",
  "KAFKA_PASSWORD": "<KAFKA_PASSWORD>",
  "KAFKA_USERNAME": "<KAFKA_USERNAME>",
  "KMS_HOST_STRING": "<KMS_HOST_STRING>",
  "MYSQL_DB": "<MYSQL_DB>",
  "MYSQL_HOST": "<MYSQL_HOST>",
  "MYSQL_PASSWORD": "<MYSQL_PASSWORD>",
  "MYSQL_PORT": "<MYSQL_PORT>",
  "MYSQL_USERNAME": "<MYSQL_USERNAME>"
}
```

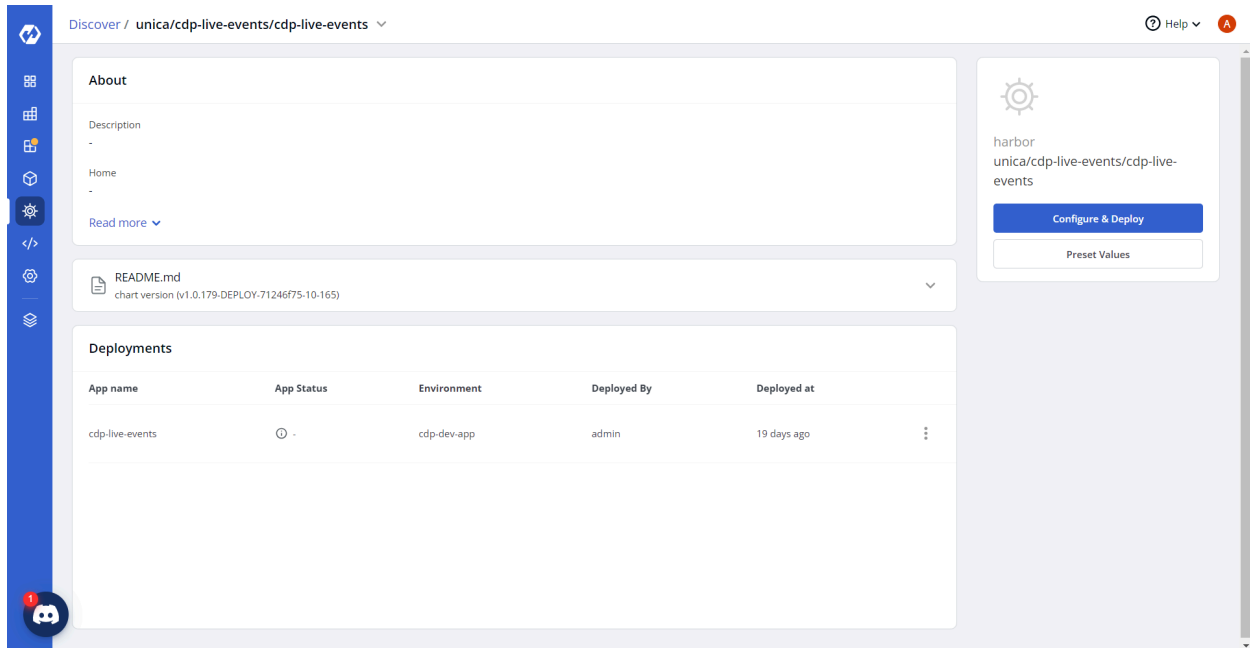
Deploying CDP Live Events

To deploy the CDP Live Events, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the bitnami/zookeeper chart to deploy.

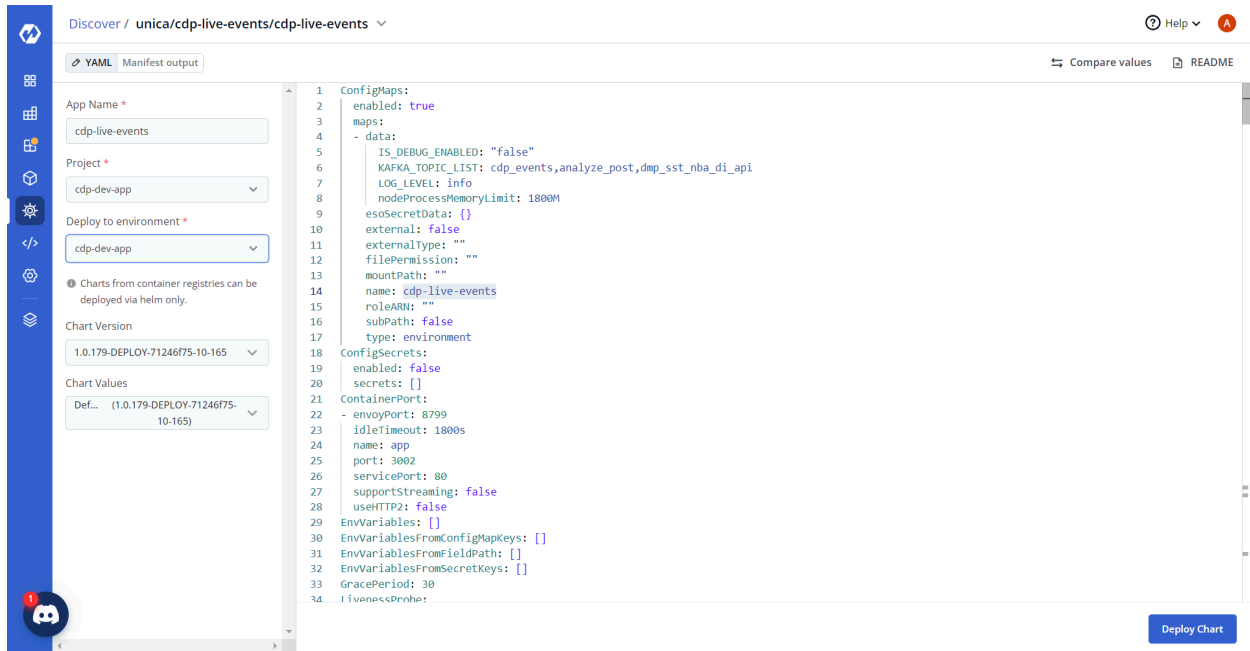


2. Now, configure and deploy the CDP Live Events charts.

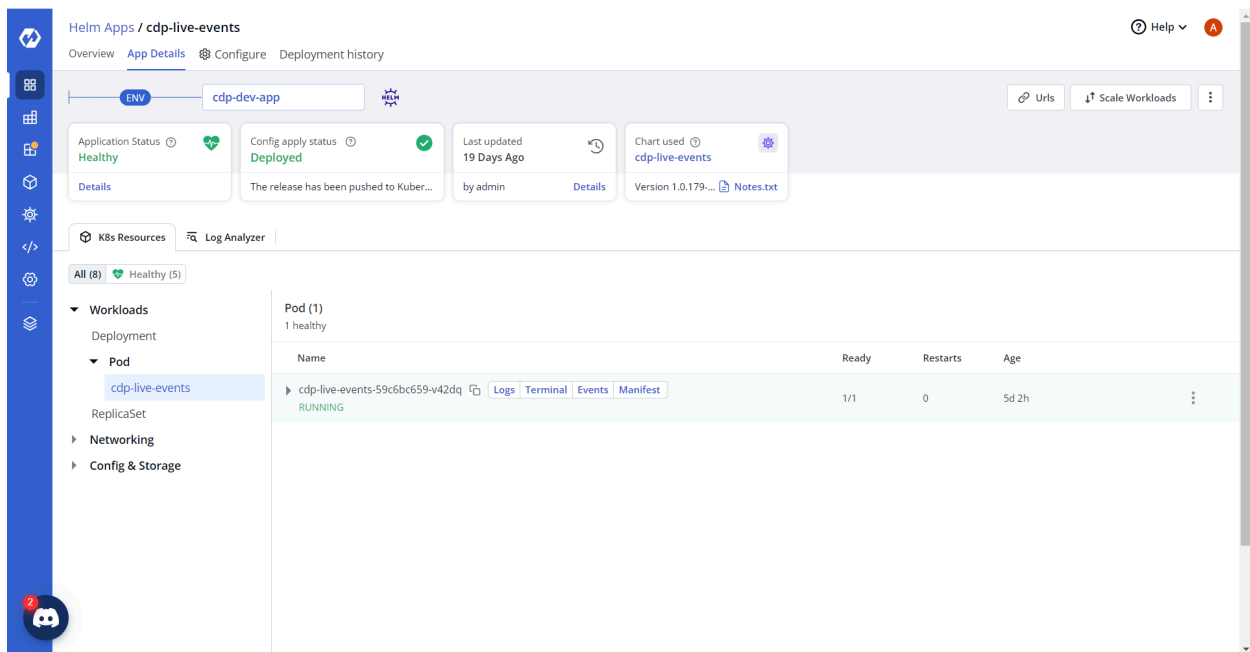


3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
IS_DEBUG_ENABLED: "false"
KAFKA_TOPIC_LIST: <cdp_events,analyze_post,dmp_sst_nba_di_api>
LOG_LEVEL: <info>
nodeProcessMemoryLimit: <1800M>
```



4. On successful deployment, validate the deployment as shown below.

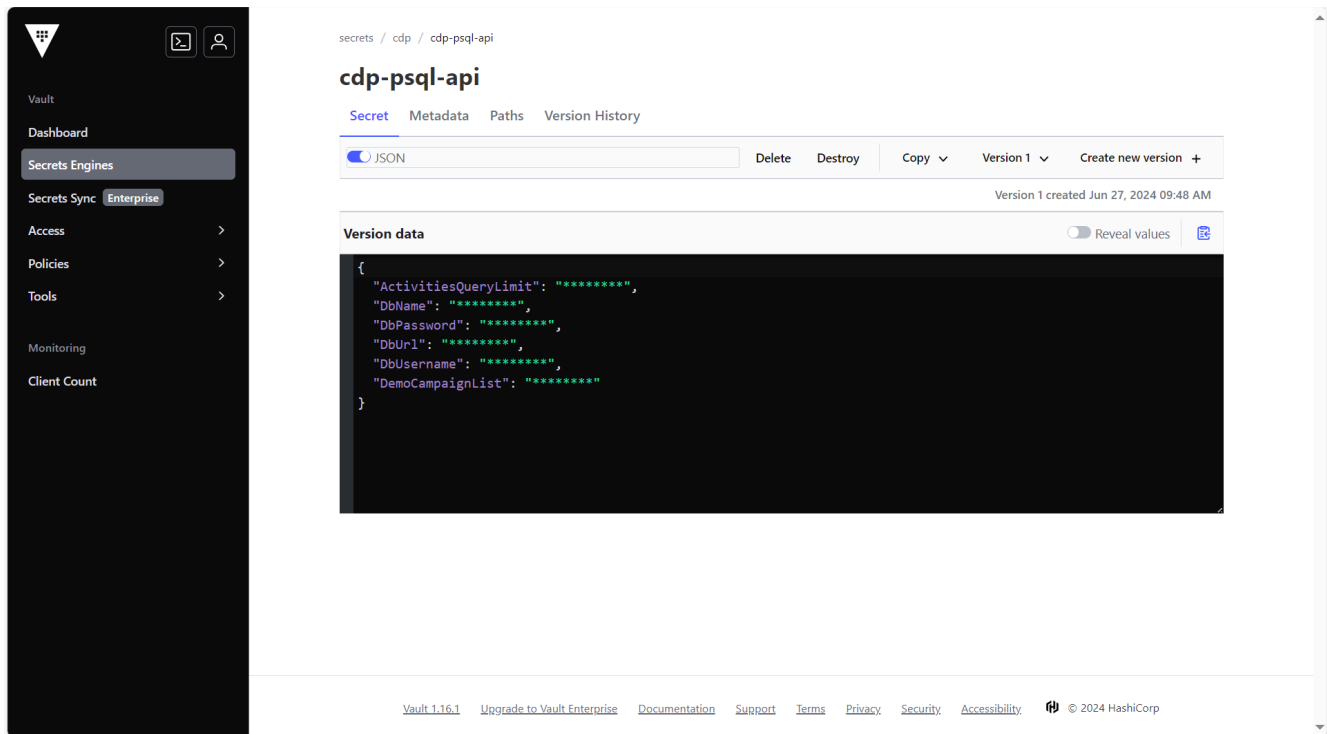


Deploying CDP PostgreSQL API

This section provides detailed instructions on how to deploy HCL CDP PSQL API using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP PSQL API.



To create the UI secret in HashiCorp Vault, follow the steps below:

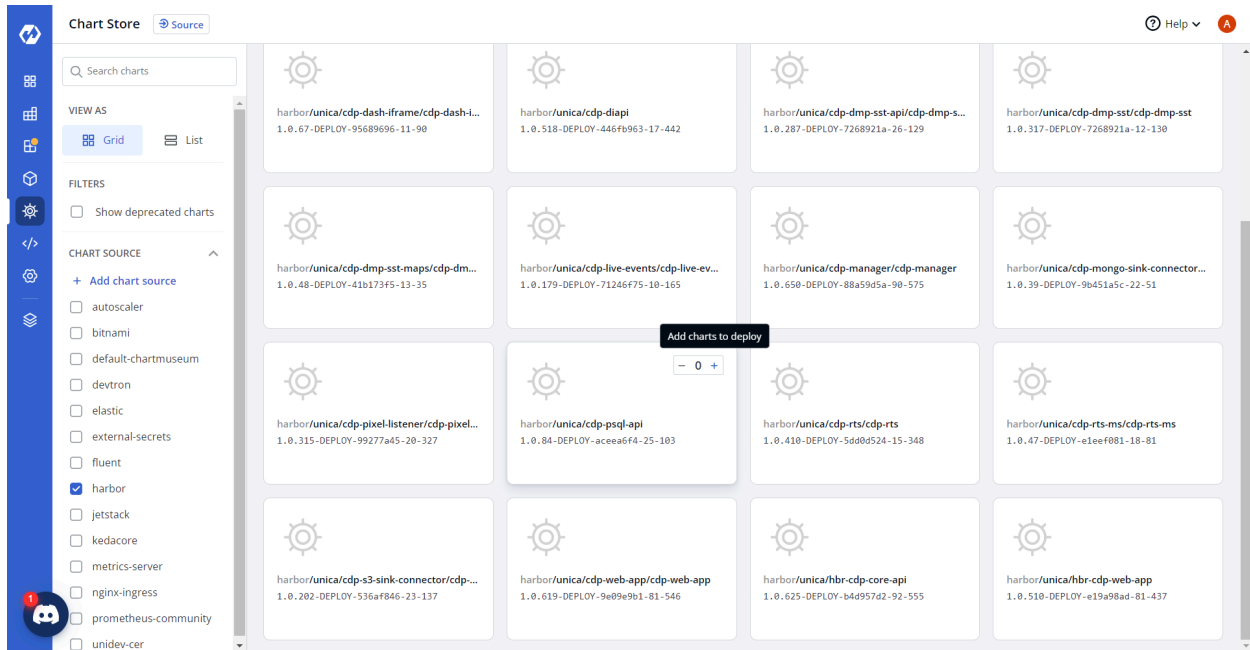
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "ActivitiesQueryLimit": "20",
  "DbName": "<DbName>",
  "DbPassword": "<DbPassword>",
  "DbUrl": "<ip:port>",
  "DbUsername": "<DbUsername>",
  "DemoCampaignList": ""
}
```

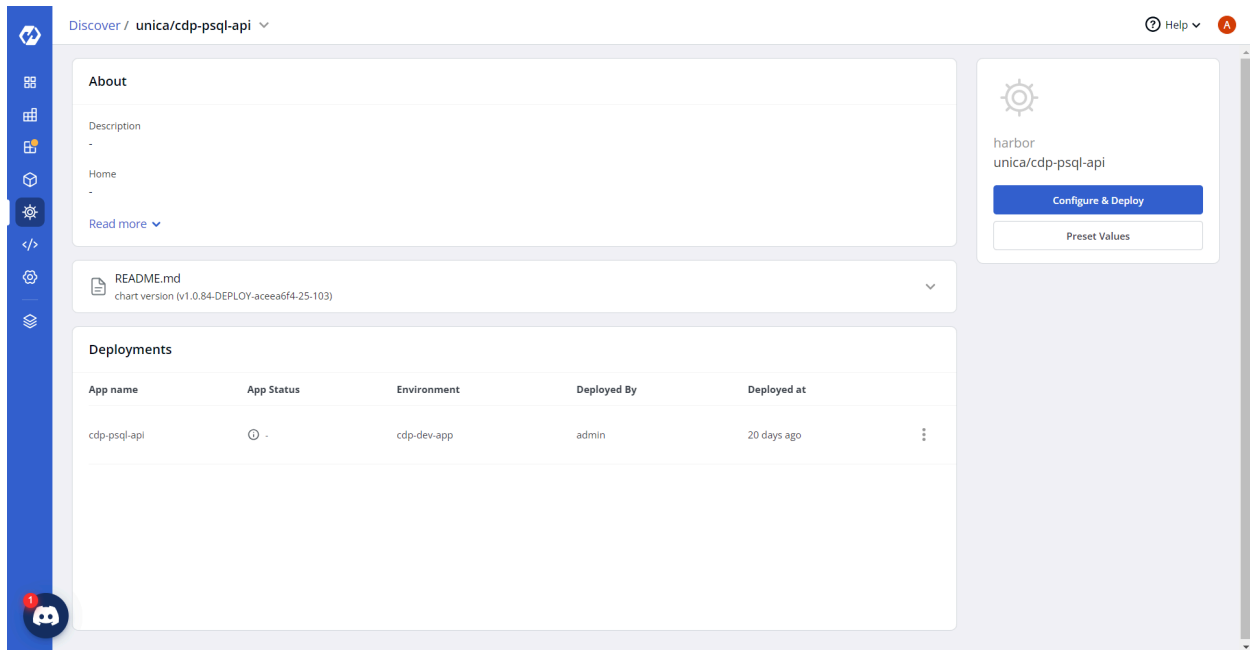
Deploying CDP PSQL API

To deploy the CDP PSQL API, follow these steps below:

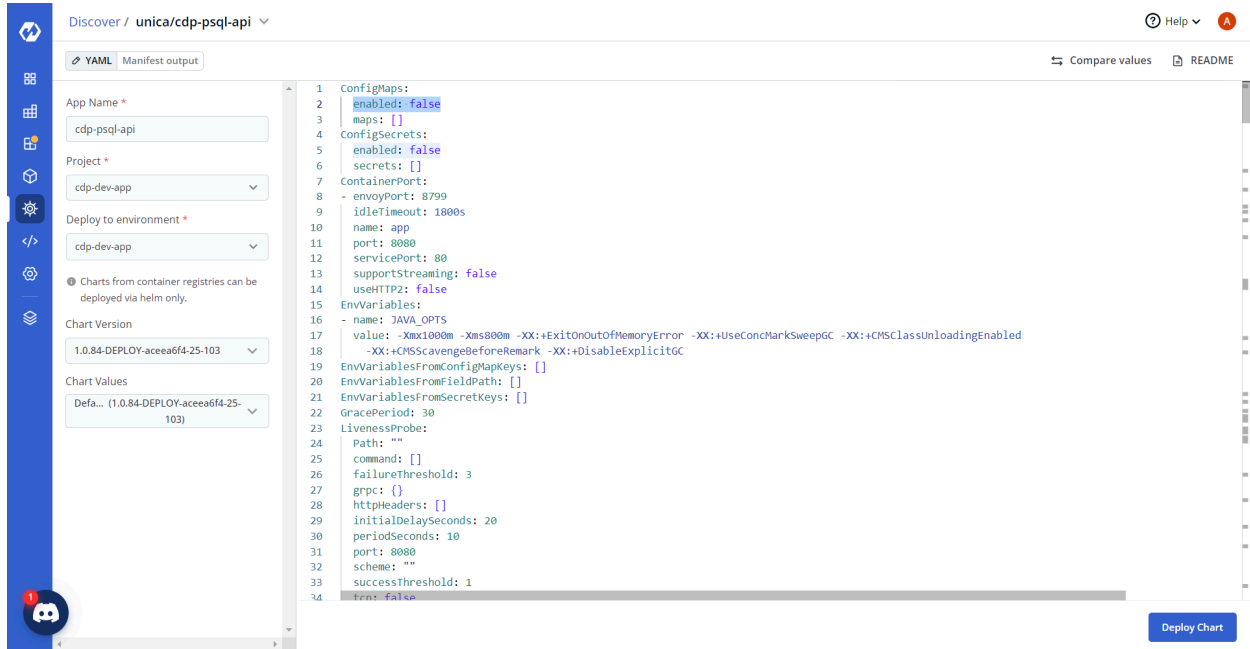
1. Navigate to the Devtron **Chart Store**, and select the `cdp-psql-api` chart to deploy.



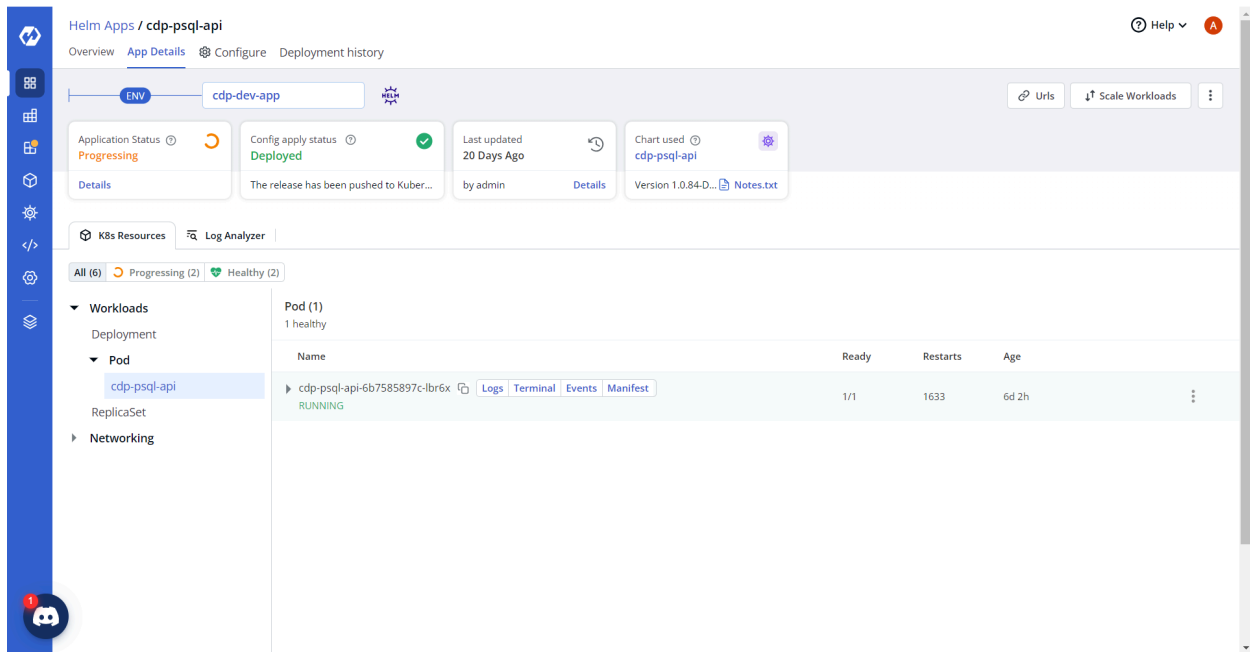
2. Now, configure and deploy the CDP PSQL API charts.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.



4. On successful deployment, validate the deployment as shown below.

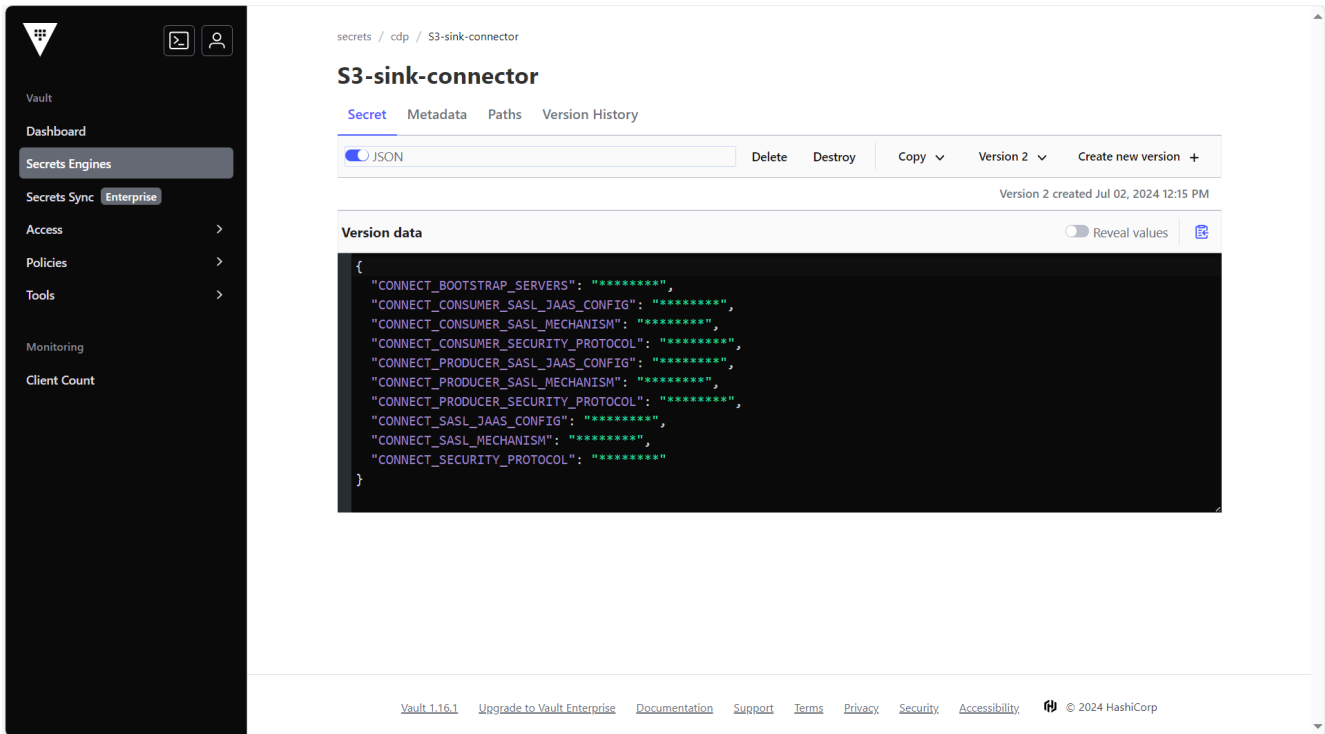


Deploying CDP S3 Sink Connector

This section provides detailed instructions on how to deploy HCL CDP S3 Sink Connector using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP S3 Sink Connector.



To create the UI secret in HashiCorp Vault, follow the steps below:

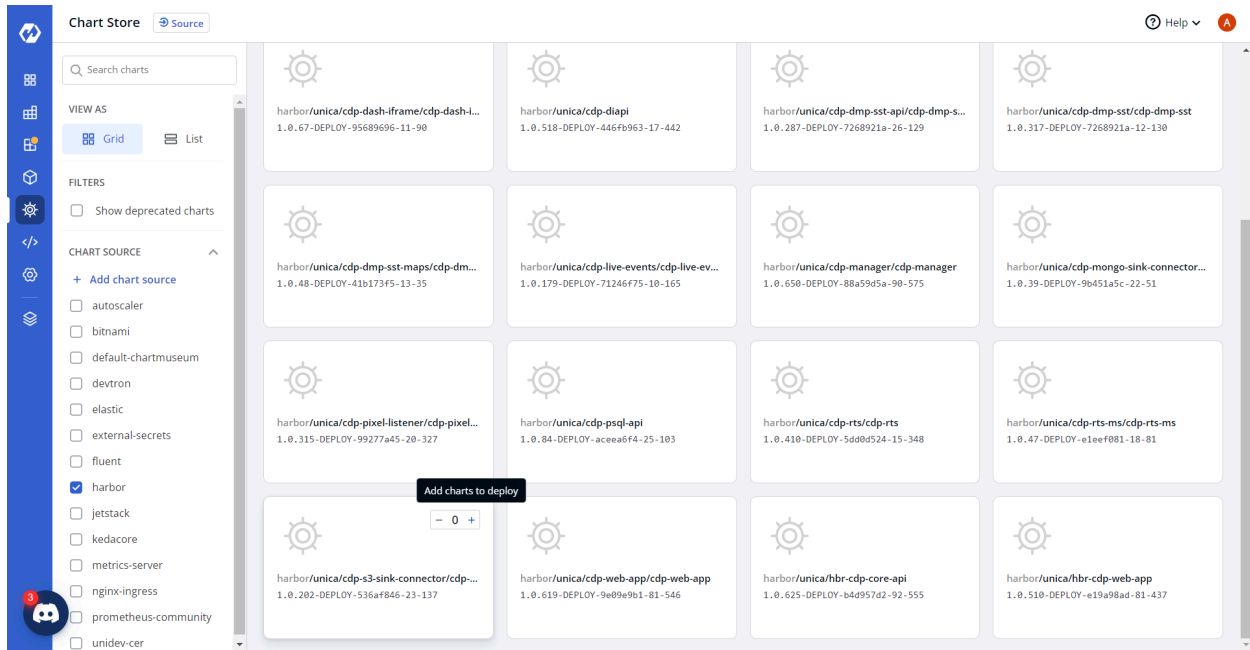
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "CONNECT_BOOTSTRAP_SERVERS": <CONNECT_BOOTSTRAP_SERVERS>,
  "CONNECT_CONSUMER_SASL_JAAS_CONFIG": <CONNECT_CONSUMER_SASL_JAAS_CONFIG>,
  "CONNECT_CONSUMER_SASL_MECHANISM": <CONNECT_CONSUMER_SASL_MECHANISM>,
  "CONNECT_CONSUMER_SECURITY_PROTOCOL": <CONNECT_CONSUMER_SECURITY_PROTOCOL>,
  "CONNECT_PRODUCER_SASL_JAAS_CONFIG": <CONNECT_PRODUCER_SASL_JAAS_CONFIG>,
  "CONNECT_PRODUCER_SASL_MECHANISM": <CONNECT_PRODUCER_SASL_MECHANISM>,
  "CONNECT_PRODUCER_SECURITY_PROTOCOL": <CONNECT_PRODUCER_SECURITY_PROTOCOL>,
  "CONNECT_SASL_JAAS_CONFIG": <CONNECT_SASL_JAAS_CONFIG>,
  "CONNECT_SASL_MECHANISM": <CONNECT_SASL_MECHANISM>,
  "CONNECT_SECURITY_PROTOCOL": <CONNECT_SECURITY_PROTOCOL>
}
```

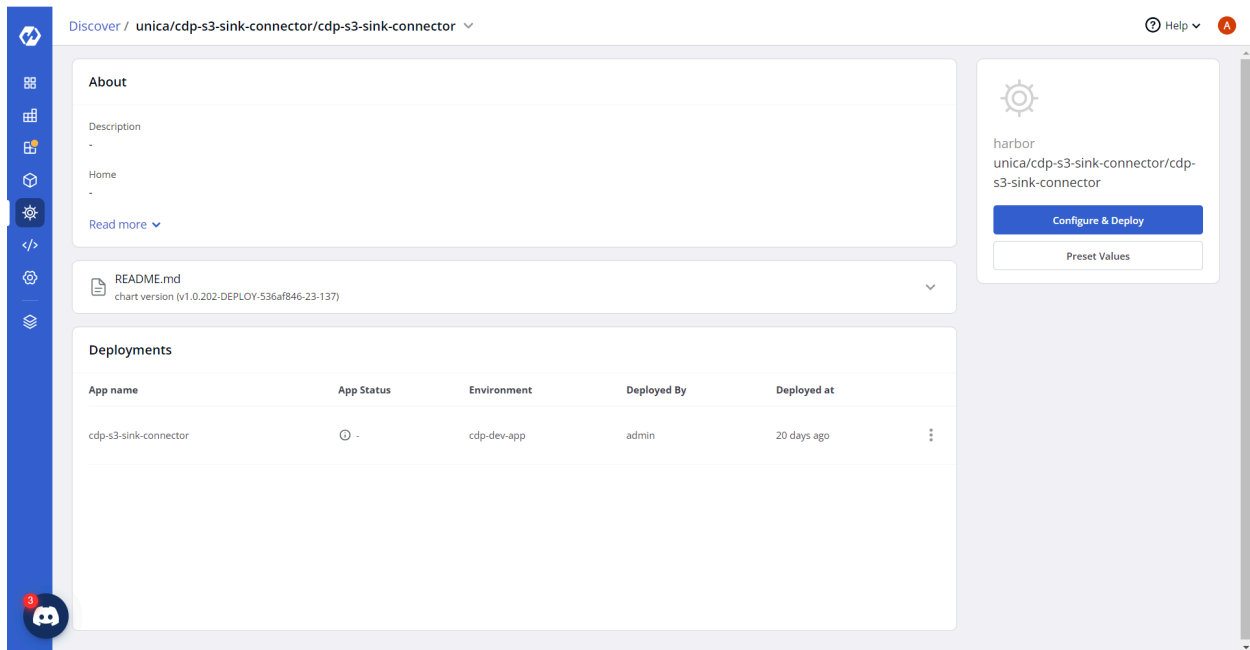
Deploying CDP S3 Sink Connector

To deploy the CDP S3 Sink Connector, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-psql-api chart to deploy.



2. Now, configure and deploy the CDP S3 Sink Connector charts.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_CONFIG_STORAGE_TOPIC: _s3-sink-connect-configs
CONNECT_CONNECTIONS_MAX_IDLE_MS: "-1"
CONNECT_GROUP_ID: s3-sink-connect
CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
CONNECT_LISTENERS: http://0.0.0.0:8083
```



```

CONNECT_LOG4J_APPENDER_STDOUT_LAYOUT_CONVERSIONPATTERN: "[%d] %p %X{connector.context}%m
(%c:%L)%n"
CONNECT_LOG4J_LOGGERS: org.apache.kafka.connect.runtime.rest=WARN,org.reflections=ERROR
CONNECT_LOG4J_ROOT_LOGLEVEL: INFO
CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_OFFSET_STORAGE_TOPIC: _s3-sink-connect-offsets
CONNECT_PLUGIN_PATH: /usr/share/java,/usr/share/confluent-hub-components
CONNECT_REPLICATION_FACTOR: "2"
CONNECT_REQUEST_TIMEOUT_MS: "20000"
CONNECT_RETRY_BACKOFF_MS: "500"
CONNECT_SSL_ENDPOINT_IDENTIFICATION_ALGORITHM: https
CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_STATUS_STORAGE_TOPIC: _s3-sink-connect-status
CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
CONNECT_VALUE_CONVERTER_BASIC_AUTH_CREDENTIALS_SOURCE: USER_INFO
CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_BASIC_AUTH_USER_INFO: ""
CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: https://aws.confluent.cloud
CONNECT_VALUE_CONVERTER_SCHEMAS_ENABLE: "true"

```

The screenshot shows the Kubernetes Dashboard interface for the `cdp-s3-sink-connector` Helm chart. The left sidebar contains navigation icons. The main panel displays the configuration for the chart, including the App Name, Project, Deploy to environment, Chart Version, and Chart Values. The right panel shows the rendered configuration as a YAML manifest.

App Name: cdp-s3-sink-connector

Project: cdp-dev-app

Deploy to environment: cdp-dev-app

Chart Version: 1.0.202-DEPLOY-536af846-23-137

Chart Values: Def... (1.0.202-DEPLOY-536af846-23-137)

ConfigMaps:

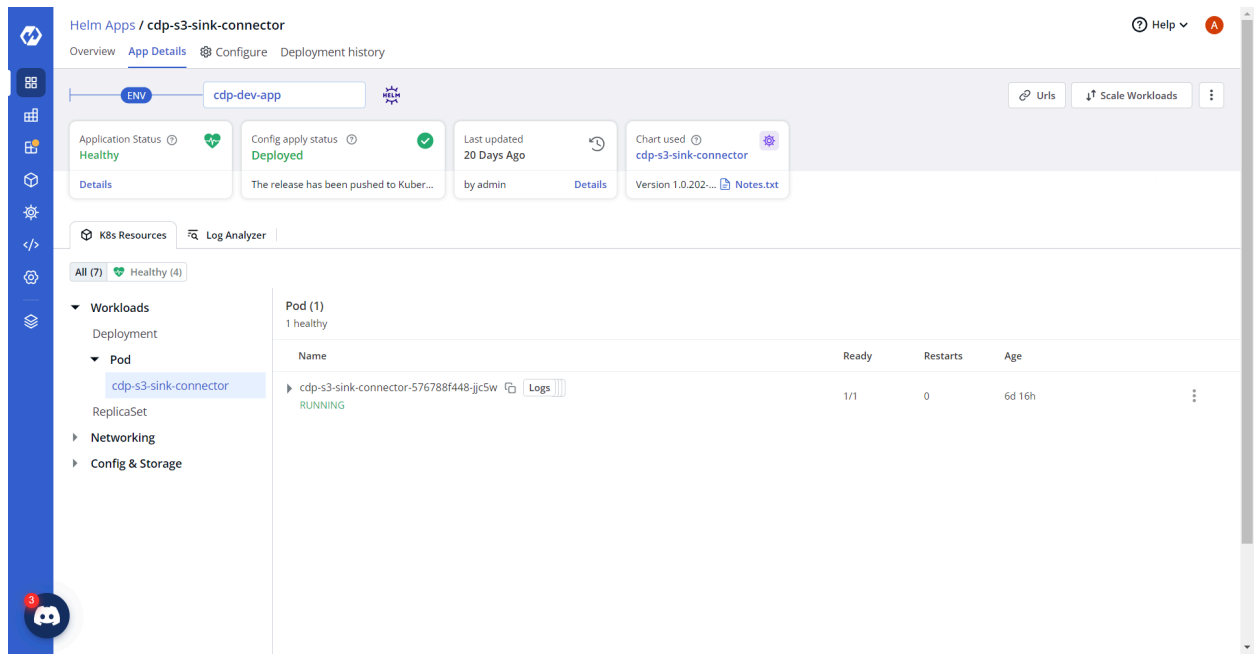
```

1 enabled: true
2 maps:
3   - data:
4     CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: "2"
5     CONNECT_CONFIG_STORAGE_TOPIC: _s3-sink-connect-configs
6     CONNECT_CONNECTIONS_MAX_IDLE_MS: "-1"
7     CONNECT_GROUP_ID: s3-sink-connect
8     CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
9     CONNECT_LISTENERS: http://0.0.0.0:8083
10    CONNECT_LOG4J_APPENDER_STDOUT_LAYOUT_CONVERSIONPATTERN: "[%d] %p %X{connector.context}%m
11    (%c:%L)%n"
12    CONNECT_LOG4J_LOGGERS: org.apache.kafka.connect.runtime.rest=WARN,org.reflections=ERROR
13    CONNECT_LOG4J_ROOT_LOGLEVEL: INFO
14    CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: "2"
15    CONNECT_OFFSET_STORAGE_TOPIC: _s3-sink-connect-offsets
16    CONNECT_PLUGIN_PATH: /usr/share/java,/usr/share/confluent-hub-components
17    CONNECT_REPLICATION_FACTOR: "2"
18    CONNECT_REQUEST_TIMEOUT_MS: "20000"
19    CONNECT_RETRY_BACKOFF_MS: "500"
20    CONNECT_SSL_ENDPOINT_IDENTIFICATION_ALGORITHM: https
21    CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: "2"
22    CONNECT_STATUS_STORAGE_TOPIC: _s3-sink-connect-status
23    CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
24    CONNECT_VALUE_CONVERTER_BASIC_AUTH_CREDENTIALS_SOURCE: USER_INFO
25    CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_BASIC_AUTH_USER_INFO: ""
26    CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: https://aws.confluent.cloud
27    CONNECT_VALUE_CONVERTER_SCHEMAS_ENABLE: "true"
28
29 esoSecretData: {}
30 external: false
31 externalType: ""
32 filePermission: ""
33 mountPath: ""
34 name: cdp-s3-sink-connector

```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

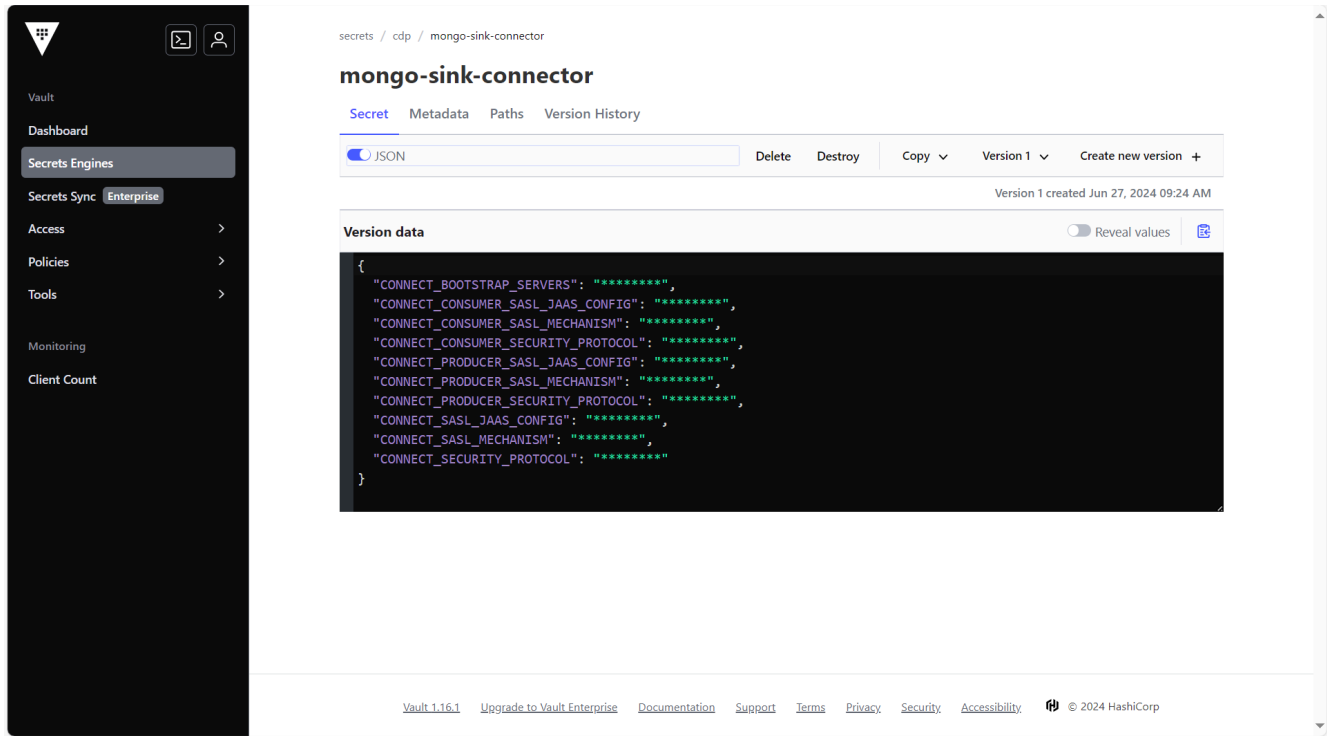


Deploying CDP Mongo Sink Connector

This section provides detailed instructions on how to deploy HCL CDP Mongo Sink Connector using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP Mongo Sink Connector.



To create the UI secret in HashiCorp Vault, follow the steps below:

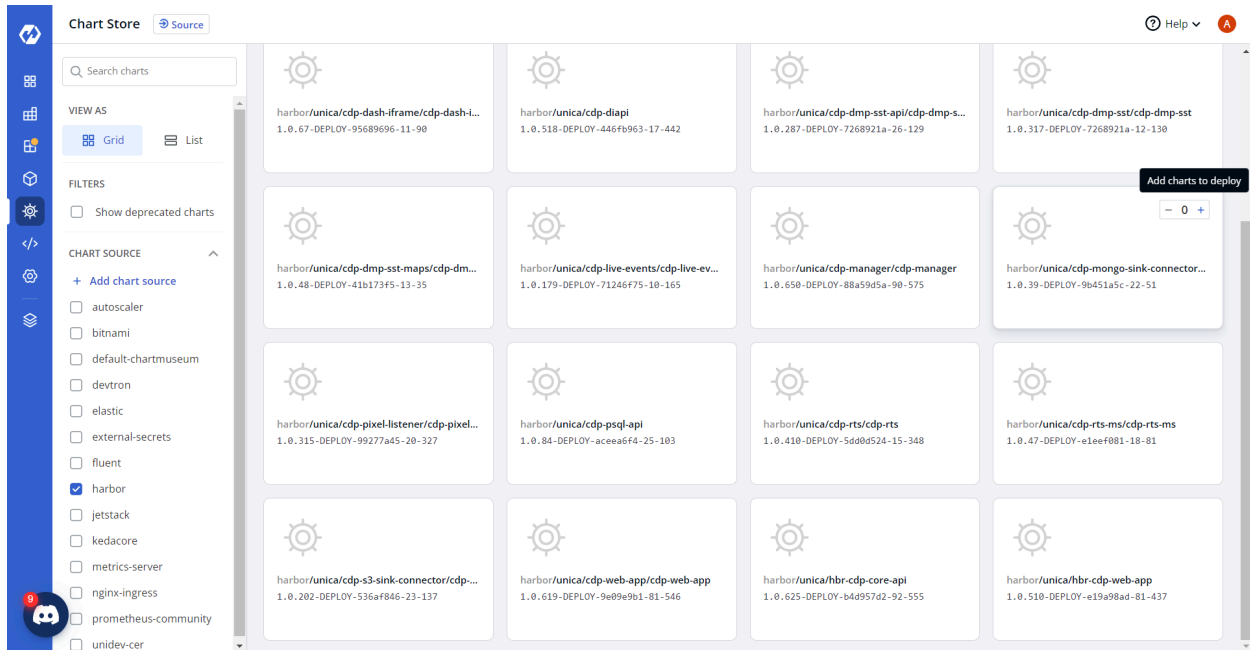
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "CONNECT_BOOTSTRAP_SERVERS": <CONNECT_BOOTSTRAP_SERVERS>,
  "CONNECT_CONSUMER_SASL_JAAS_CONFIG": <CONNECT_CONSUMER_SASL_JAAS_CONFIG>,
  "CONNECT_CONSUMER_SASL_MECHANISM": <CONNECT_CONSUMER_SASL_MECHANISM>,
  "CONNECT_CONSUMER_SECURITY_PROTOCOL": <CONNECT_CONSUMER_SECURITY_PROTOCOL>,
  "CONNECT_PRODUCER_SASL_JAAS_CONFIG": <CONNECT_PRODUCER_SASL_JAAS_CONFIG>,
  "CONNECT_PRODUCER_SASL_MECHANISM": <CONNECT_PRODUCER_SASL_MECHANISM>,
  "CONNECT_PRODUCER_SECURITY_PROTOCOL": <CONNECT_PRODUCER_SECURITY_PROTOCOL>,
  "CONNECT_SASL_JAAS_CONFIG": <CONNECT_SASL_JAAS_CONFIG>,
  "CONNECT_SASL_MECHANISM": <CONNECT_SASL_MECHANISM>,
  "CONNECT_SECURITY_PROTOCOL": <CONNECT_SECURITY_PROTOCOL>
}
```

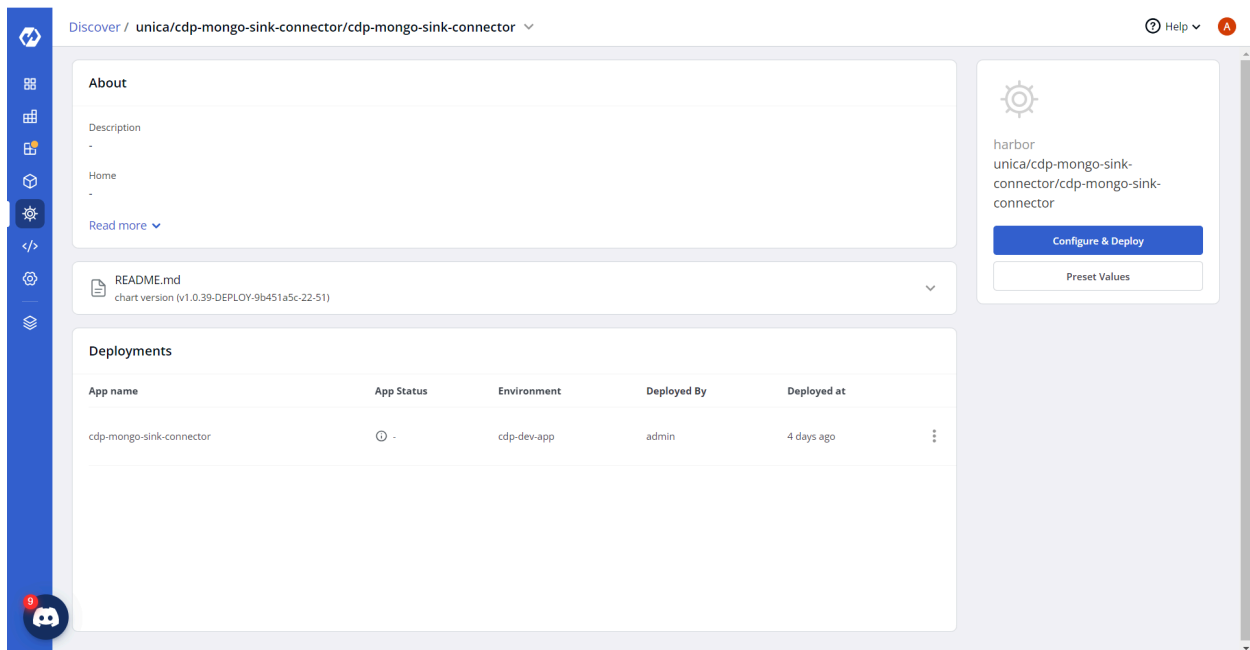
Deploying CDP Mongo Sink Connector

To deploy the CDP Mongo Sink Connector, follow these steps below:

1. Navigate to the Devtron **Chart Store**, and select the cdp-mongo-sink-connector chart to deploy.



2. Now, configure and deploy the CDP Mongo Sink Connector.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.

```
CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_CONFIG_STORAGE_TOPIC: _kafka-segdb-connect-configs
CONNECT_GROUP_ID: kafka-segdb-connect
CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
CONNECT_LISTENERS: http://0.0.0.0:8083
CONNECT_LOG4J_APPENDER_STDOUT_LAYOUT_CONVERSIONPATTERN: "[%d] %p %X{connector.context}%m
```

```
(%c:%L)%n'
CONNECT_LOG4J_LOGGERS: org.apache.kafka.connect.runtime.rest=WARN,org.reflections=ERROR
CONNECT_LOG4J_ROOT_LOGLEVEL: INFO
CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_OFFSET_STORAGE_TOPIC: _kafka-segdb-connect-offsets
CONNECT_PLUGIN_PATH: /usr/share/java,/usr/share/confluent-hub-components
CONNECT_REPLICATION_FACTOR: "2"
CONNECT_REQUEST_TIMEOUT_MS: "20000"
CONNECT_RETRY_BACKOFF_MS: "500"
CONNECT_SSL_ENDPOINT_IDENTIFICATION_ALGORITHM: https
CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: "2"
CONNECT_STATUS_STORAGE_TOPIC: _kafka-segdb-connect-status
CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
CONNECT_VALUE_CONVERTER_BASIC_AUTH_CREDENTIALS_SOURCE: USER_INFO
CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_BASIC_AUTH_USER_INFO: ""
CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: https://aws.confluent.cloud
CONNECT_VALUE_CONVERTER_SCHEMAS_ENABLE: "true"
```

Discover / unica/cdp-mongo-sink-connector/cdp-mongo-sink-connector

YAML Manifest output

App Name *

cdp-mongo-sink-connector

Project *

cdp-dev-app

Deploy to environment *

cdp-dev-app

Charts from container registries can be deployed via helm only.

Chart Version

1.0.39-DEPLOY-9b451a5c-22-51

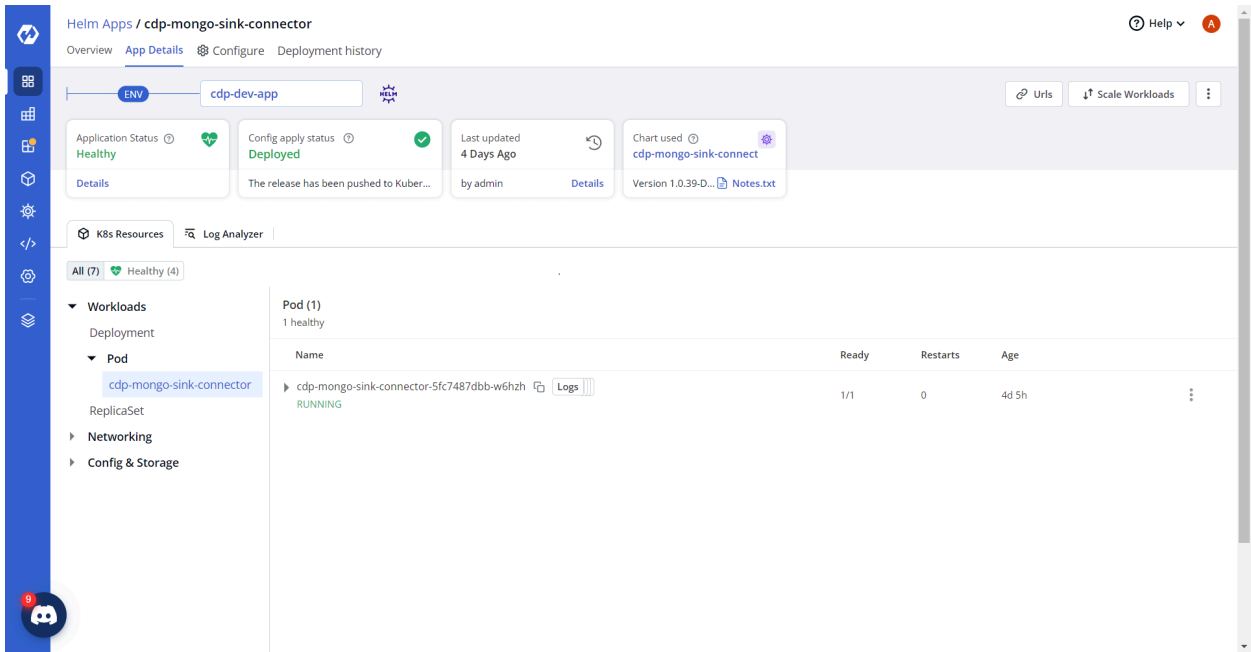
Chart Values

Defa... (1.0.39-DEPLOY-9b451a5c-22-51)

```
1 ConfigMaps:
2   enabled: true
3   maps:
4     - data:
5         CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: "2"
6         CONNECT_CONFIG_STORAGE_TOPIC: _kafka-segdb-connect-configs
7         CONNECT_GROUP_ID: kafka-segdb-connect
8         CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
9         CONNECT_LISTENERS: http://0.0.0.0:8083
10        CONNECT_LOG4J_APPENDER_STDOUT_LAYOUT_CONVERSIONPATTERN: "[%d] %p %X(connector.context)%n'
11          (%c:%L)%n'
12        CONNECT_LOG4J_LOGGERS: org.apache.kafka.connect.runtime.rest=WARN,org.reflections=ERROR
13        CONNECT_LOG4J_ROOT_LOGLEVEL: INFO
14        CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: "2"
15        CONNECT_OFFSET_STORAGE_TOPIC: _kafka-segdb-connect-offsets
16        CONNECT_PLUGIN_PATH: /usr/share/java,/usr/share/confluent-hub-components
17        CONNECT_REPLICATION_FACTOR: "2"
18        CONNECT_REQUEST_TIMEOUT_MS: "20000"
19        CONNECT_RETRY_BACKOFF_MS: "500"
20        CONNECT_SSL_ENDPOINT_IDENTIFICATION_ALGORITHM: https
21        CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: "2"
22        CONNECT_STATUS_STORAGE_TOPIC: _kafka-segdb-connect-status
23        CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
24        CONNECT_VALUE_CONVERTER_BASIC_AUTH_CREDENTIALS_SOURCE: USER_INFO
25        CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_BASIC_AUTH_USER_INFO: ""
26        CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: https://aws.confluent.cloud
27        CONNECT_VALUE_CONVERTER_SCHEMAS_ENABLE: "true"
28      esoSecretData: {}
29      external: false
30      externalType: ""
31      filePermission: ""
32      mountPath: ""
33      name: cdp-mongo-sink-connector
34      roleARN: ""
```

Deploy Chart

4. On successful deployment, validate the deployment as shown below.

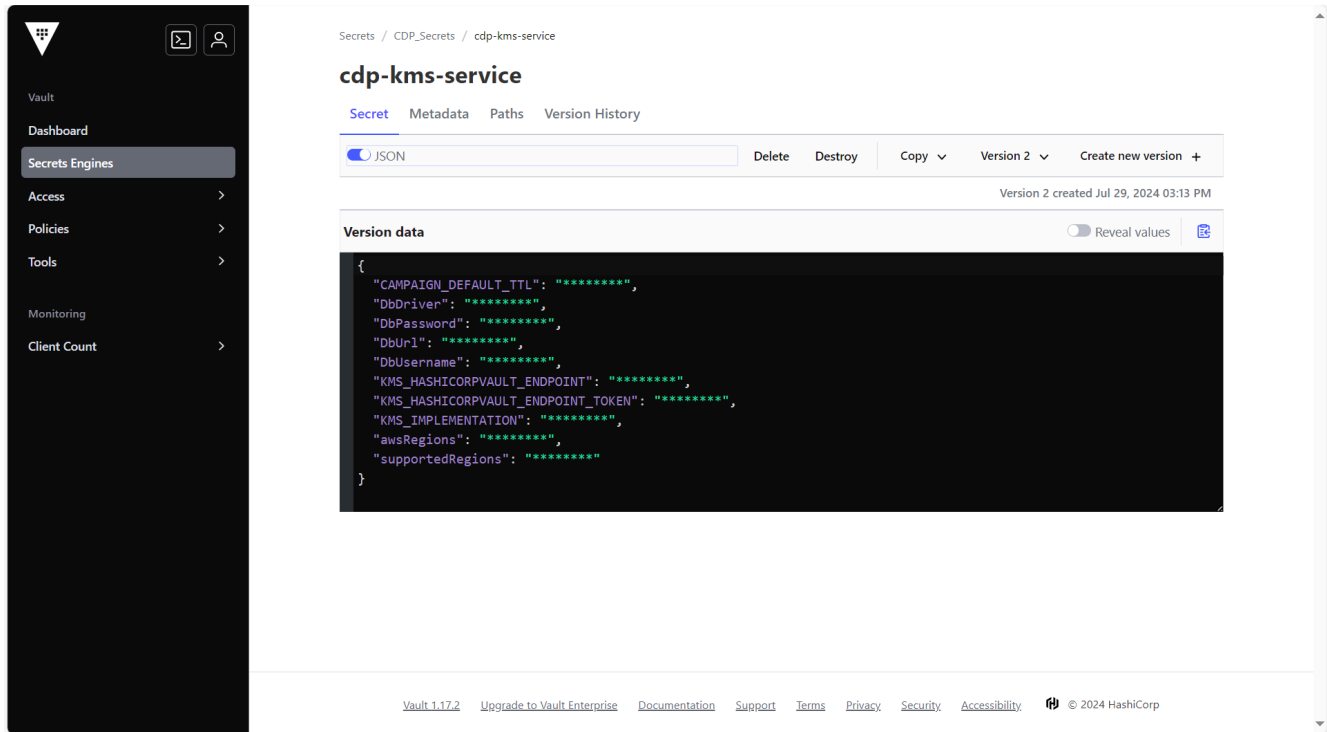


Deploying CDP KMS Service

This section provides detailed instructions on how to deploy HCL CDP KMS Service using the Devtron in the OpenShift.

Prerequisites:

Make sure to create UI secret with required data in HashiCorp vault before deploying CDP KMS Service.



To create the UI secret in HashiCorp Vault, follow the steps below:

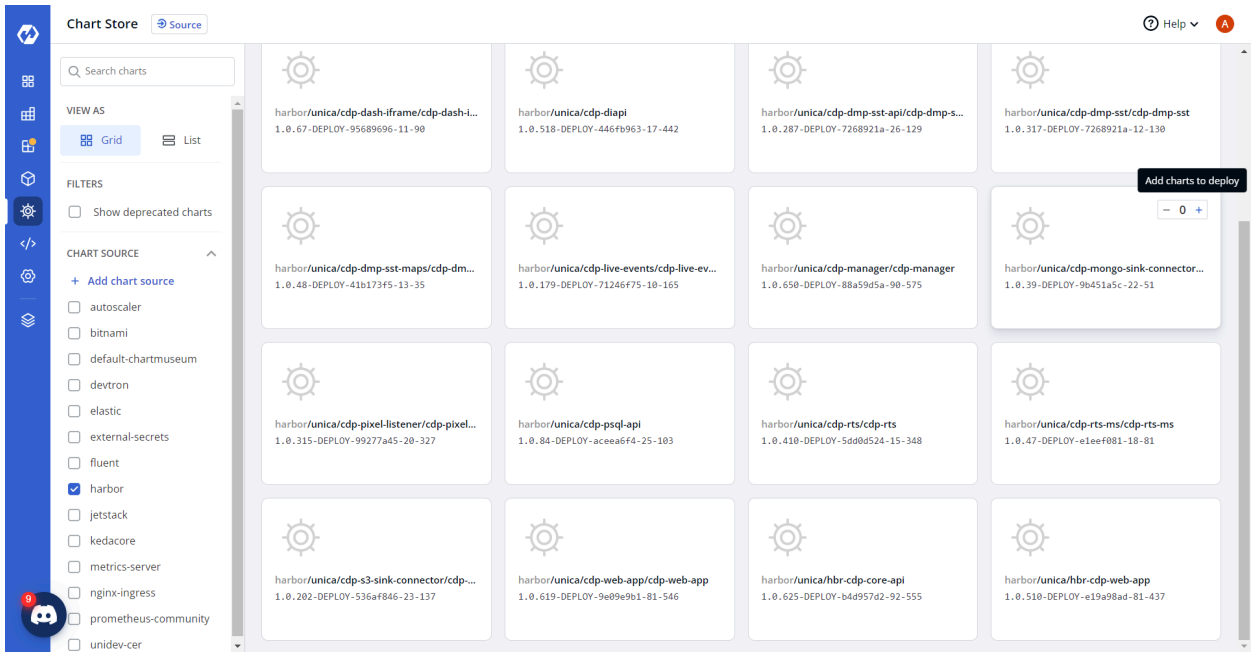
1. Create a UI secret sample key and value in the UI secret, and update ConfigMaps data with actual values.

```
{
  "CAMPAIGN_DEFAULT_TTL": "21600",
  "DbDriver": "<DbDriver>",
  "DbPassword": "<DbPassword>",
  "DbUrl": "<DbUrl>",
  "DbUsername": "<DbUsername>",
  "KMS_HASHICORPVAULT_ENDPOINT": "<KMS_HASHICORPVAULT_ENDPOINT>",
  "KMS_HASHICORPVAULT_ENDPOINT_TOKEN": "<KMS_HASHICORPVAULT_ENDPOINT_TOKEN>",
  "KMS_IMPLEMENTATION": "HashiCorpVault",
  "awsRegions": "ap-south-1",
  "supportedRegions": "aps1"
}
```

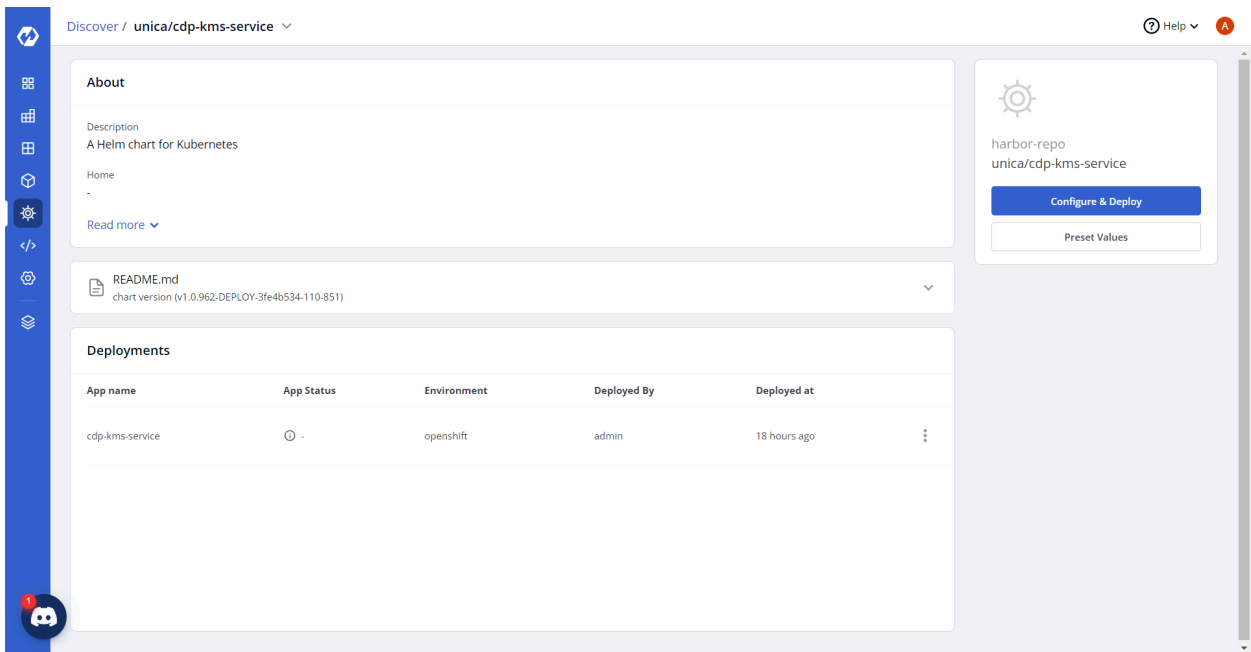
Deploying CDP KMS Service

To deploy the CDP KMS Service, follow these steps below:

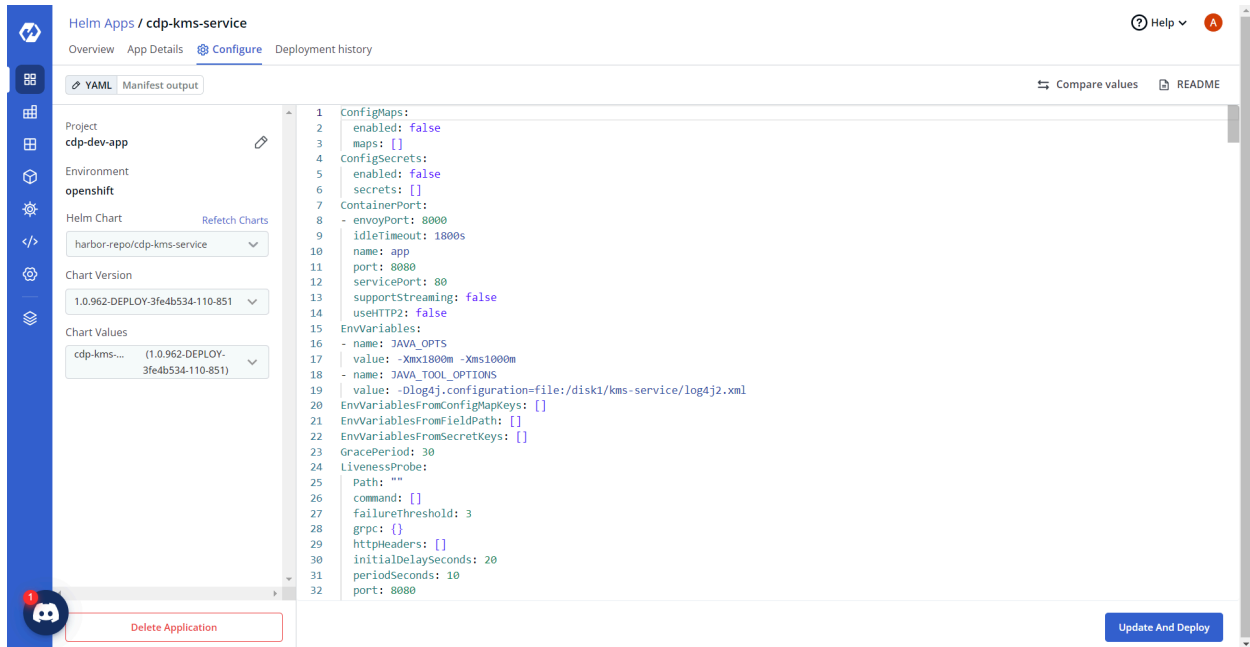
1. Navigate to the Devtron **Chart Store**, and select the `cdp-mongo-sink-connector` chart to deploy.



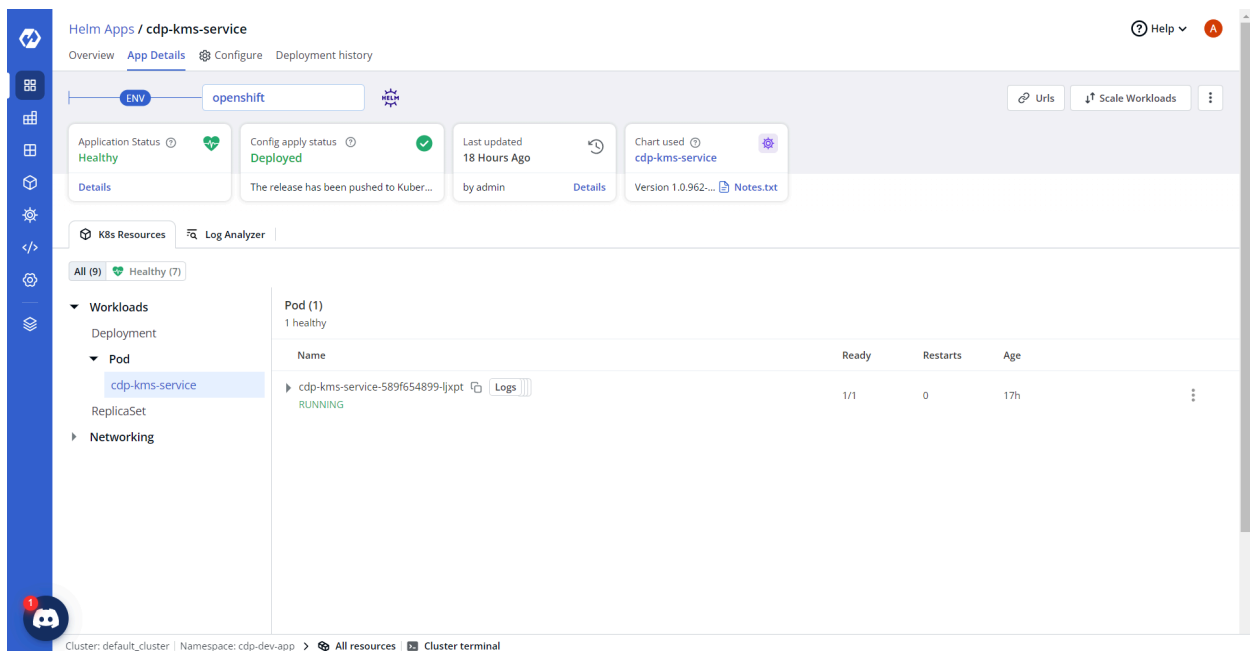
2. Now, configure and deploy the CDP KMS Service.



3. In the **YAML** section, update the ConfigMap using below details, and deploy the charts.



4. On successful deployment, validate the deployment as shown below.

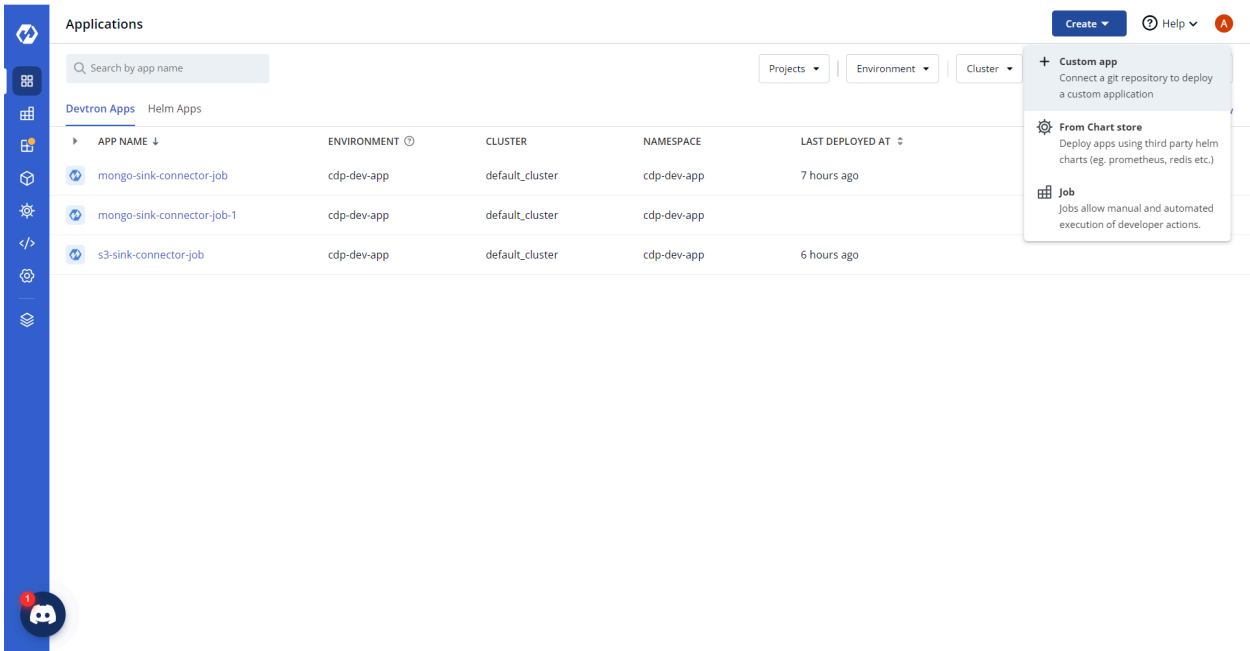


Deploying CDP Mongo Connector Sink Job

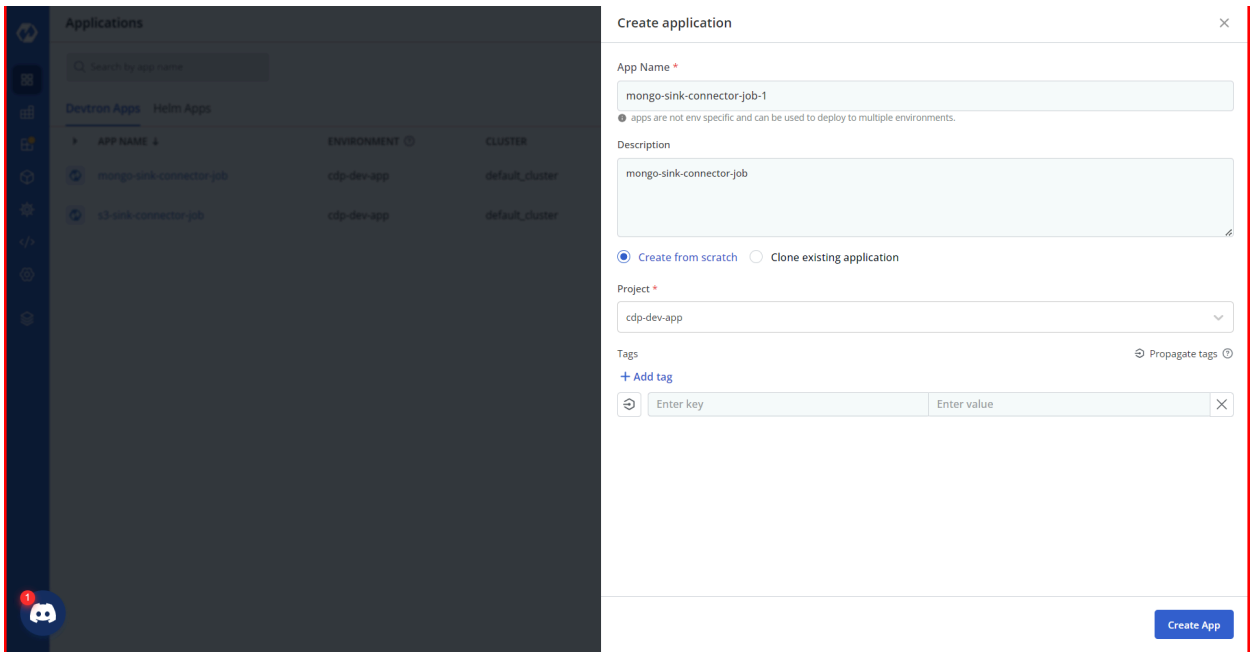
This section provides detailed instructions on how to deploy HCL CDP Mongo Connector Sink Job using the Devtron in the OpenShift.

Creating custom app

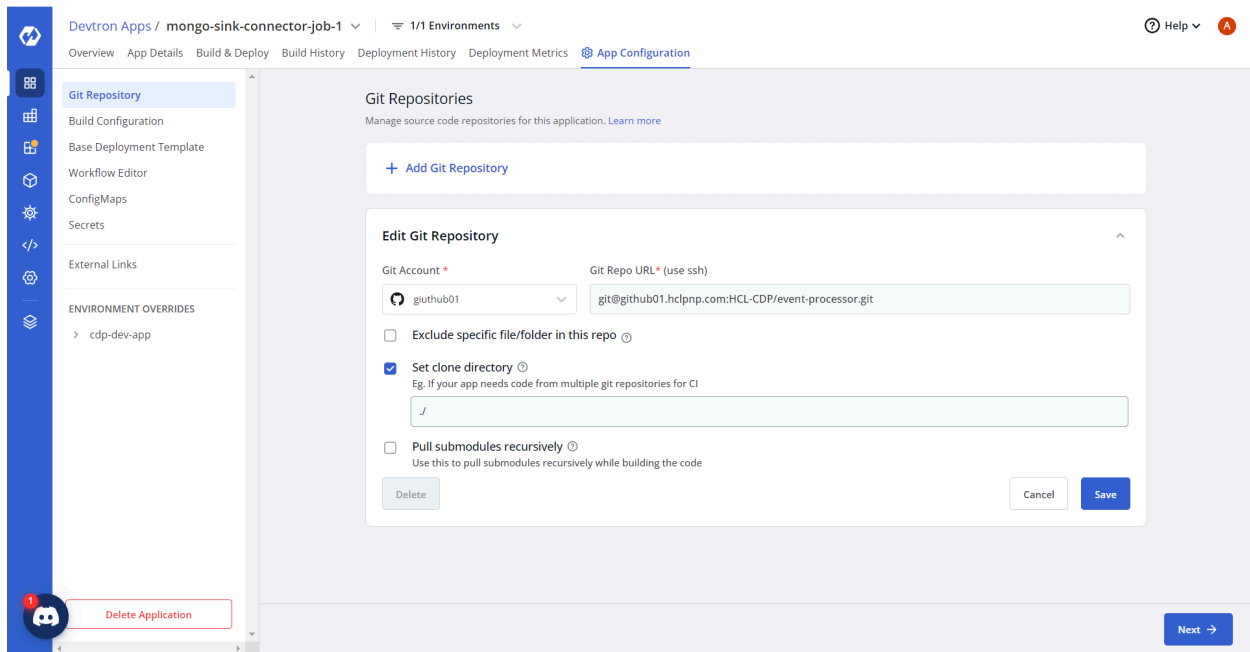
1. In the devtron console, select **Applications**, and click **Create > Custom app**.



2. In the **Create Application** section, enter application name as "mongo-sink-connector-job" and select a Project to store the application.



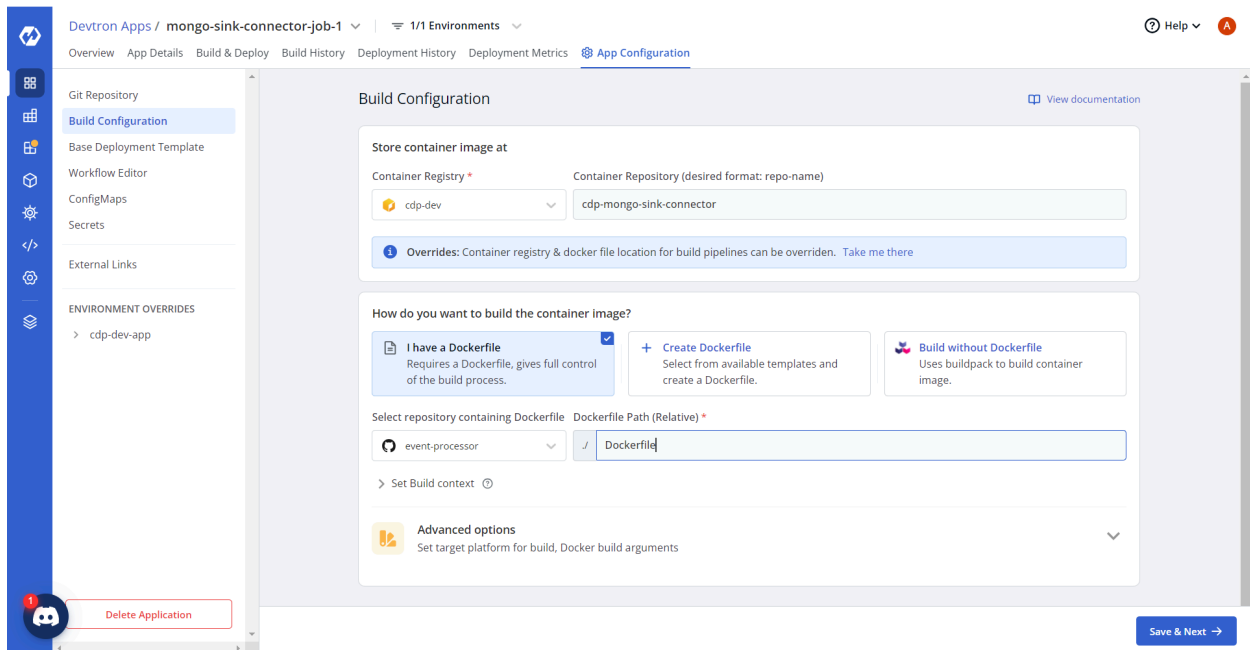
- Click **Create App** to create the application. Now, click **App Configuration** tab, and update source code repositories for the application.



Building custom app

After successfully creating the custom app, build the custom app with docker image, and configure a workflow for the custom app.

1. In the Devtron Apps page, click **Build Configurations**. In the **App Configuration** tab, select **I have a Dockerfile** option and update the dockerfile path.



2. Click **Save & Next**, and in the **Base Deployment Template** section, update chart type and Mongo Connector Sink URLs in the base deployment template as shown below.

Deployment Template

```
ContainerPort:
- envoyPort: 8799
  idleTimeout: 1800s
  name: app
  port: 8080
  servicePort: 80
  supportStreaming: true
  useHTTP2: true
EnvVariables: []
GracePeriod: 30
MaxSurge: 1
MaxUnavailable: 0
MinReadySeconds: 60
Spec:
  Affinity:
    Key: null
    Values: nodes
    key: ""
  args:
    enabled: true
    value:
      - /bin/sh
      - -c
      - >
        until $(curl --output /dev/null --silent --head --fail
        http://cdp-mongo-sink-connector-service.cdp-dev-app.svc.cluster.local:80);
```

```

do
    echo "Waiting for Kafka Connect API..."
    sleep 5
done

echo "Posting Kafka Connect configuration..."

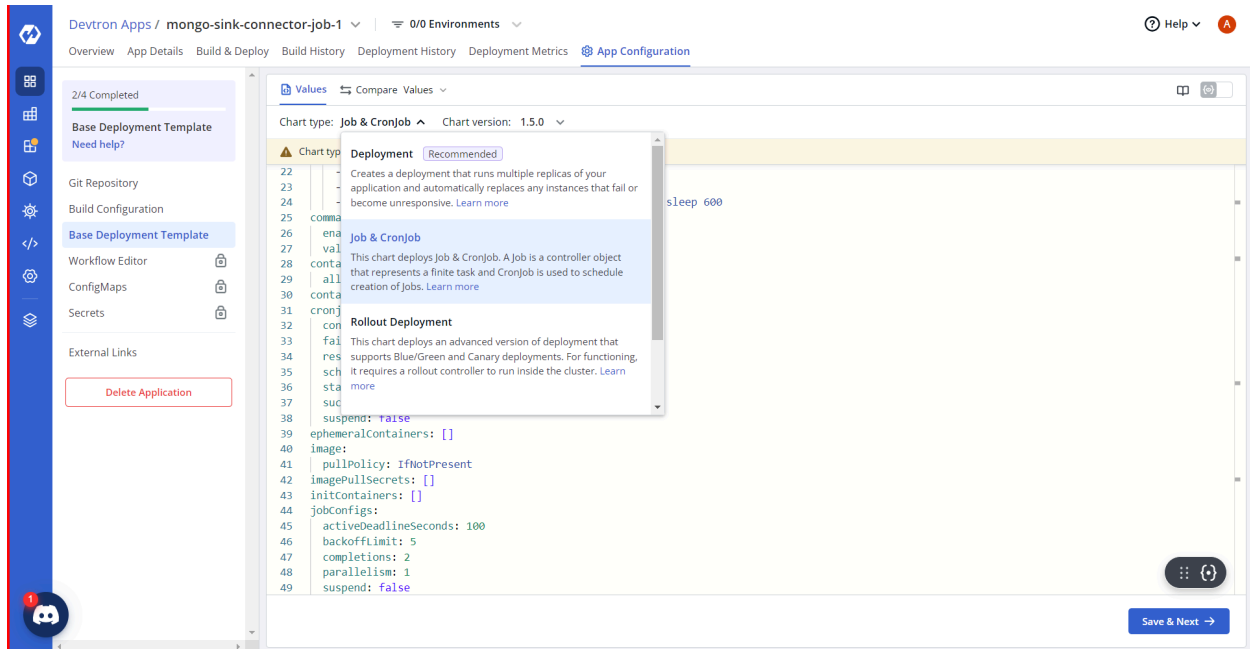
curl -X POST -H "Content-Type: application/json" --data @/data/${APP_NAME}
http://cdp-mongo-sink-connector-service.cdp-dev-app.svc.cluster.local:80/connectors
command:
  enabled: false
  value: []
containerSecurityContext:
  allowPrivilegeEscalation: false
containers: []
cronjobConfigs:
  concurrencyPolicy: Allow
  failedJobsHistoryLimit: 1
  restartPolicy: OnFailure
  schedule: "* * * * *"
  startingDeadlineSeconds: 100
  successfulJobsHistoryLimit: 3
  suspend: false
ephemeralContainers: []
image:
  pullPolicy: IfNotPresent
imagePullSecrets: []
initContainers: []
jobConfigs:
  activeDeadlineSeconds: 100
  backoffLimit: 5
  completions: 1
  parallelism: 1
  suspend: false
kedaAutoscaling:
  envSourceContainerName: ""
  failedJobsHistoryLimit: 5
  maxReplicaCount: 2
  minReplicaCount: 1
  pollingInterval: 30
  rolloutStrategy: default
  scalingStrategy:
    customScalingQueueLengthDeduction: 1
    customScalingRunningJobPercentage: "0.5"
    multipleScalersCalculation: max
    pendingPodConditions:
      - Ready
      - PodScheduled
      - AnyOtherCustomPodCondition
    strategy: custom
  successfulJobsHistoryLimit: 5
triggerAuthentication:
  enabled: false
  name: ""
  spec: {}
triggers:
  - authenticationRef: {}
    metadata:

```

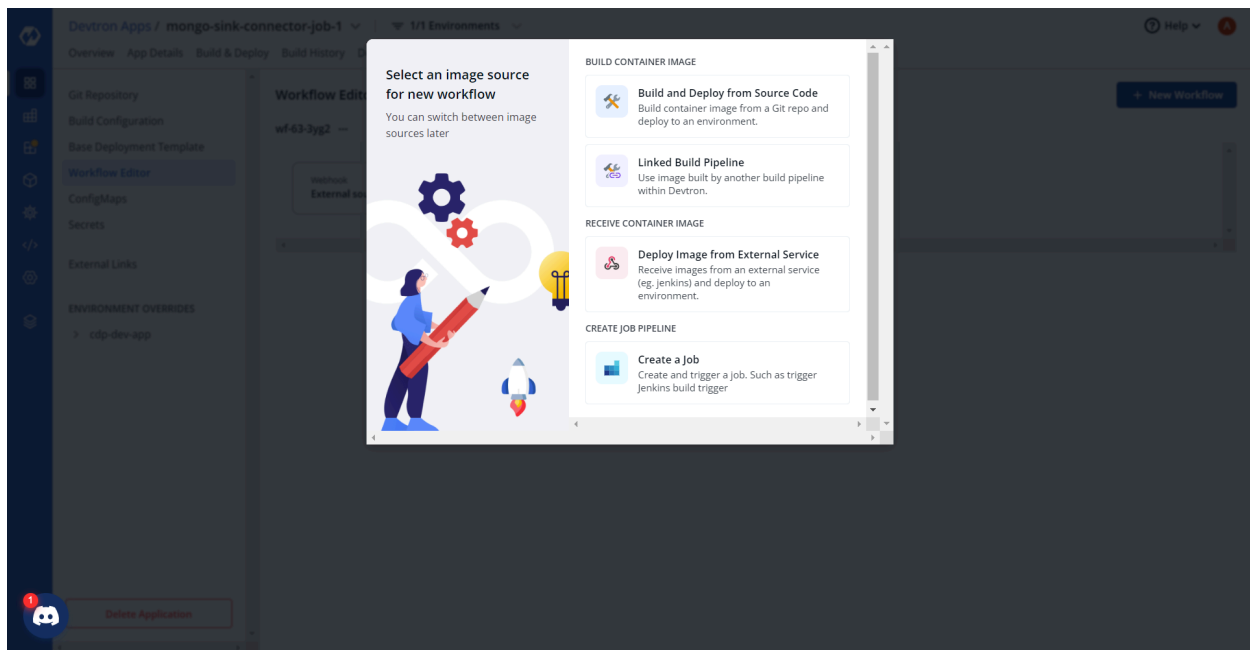
```

      host: RabbitMqHost
      queueLength: "5"
      queueName: hello
      type: rabbitmq
kind: Job
pauseForSecondsBeforeSwitchActive: 30
podAnnotations: {}
podLabels: {}
podSecurityContext: {}
prometheus:
  release: monitoring
rawYaml: []
readinessGates: []
resources:
  limits:
    cpu: "0.05"
    memory: 50Mi
  requests:
    cpu: "0.01"
    memory: 10Mi
secret:
  data: {}
  enabled: false
server:
  deployment:
    image: ""
    image_tag: 1-95af053
service:
  annotations: {}
  enabled: false
  type: ClusterIP
servicemonitor:
  additionalLabels: {}
setHostnameAsFQDN: false
shareProcessNamespace: false
tolerations: []
topologySpreadConstraints: []
volumeMounts: []
volumes: []
waitForSecondsBeforeScalingDown: 30

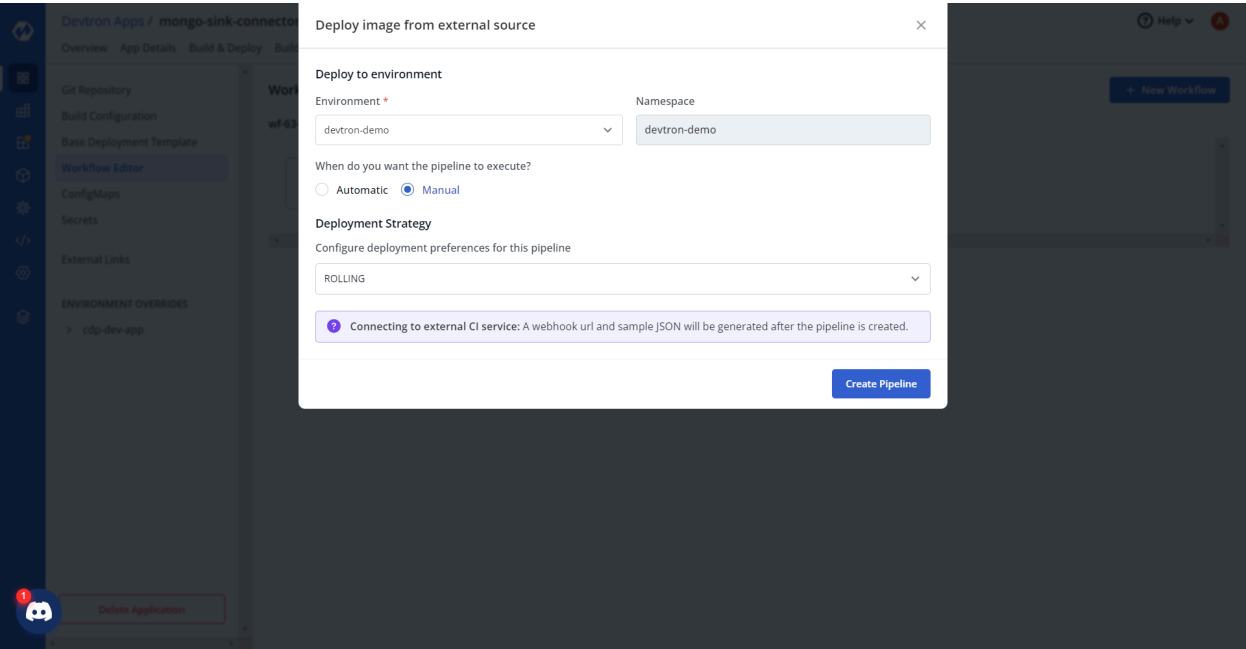
```



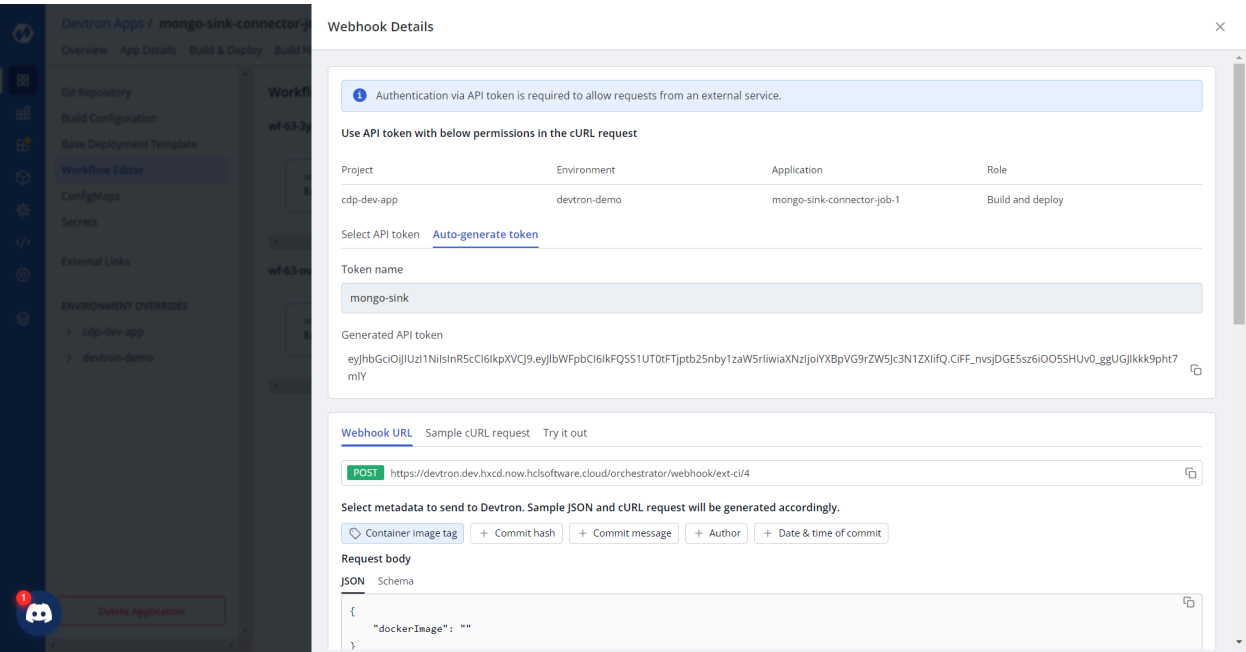
3. Click **Save & Next**. In the Workflow Editor section, add new workflow, and select the **Deploy Image from External Source** option.



4. Configure required parameters, and click **Create pipeline** at the right bottom corner of the Deploy image from external source section.

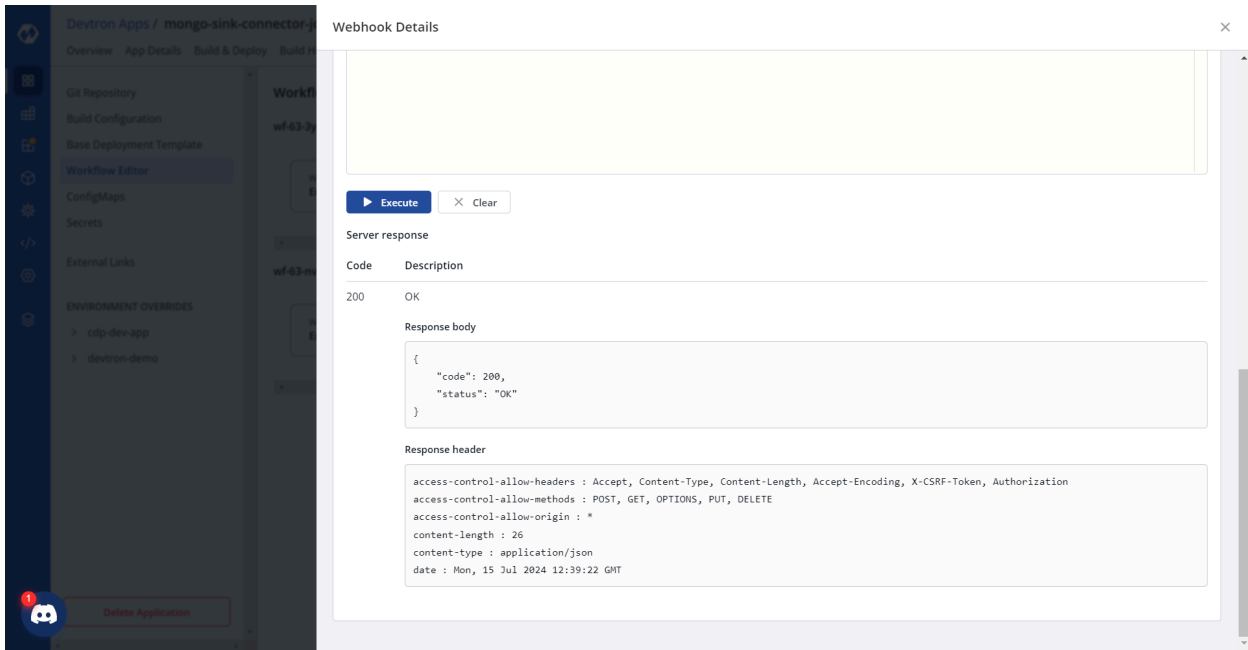


5. In the Workflow Editor, click **Edit Workflow**, and in the **Webhook Details** section, click the **Auto Generate Token** tab, and copy the token details.



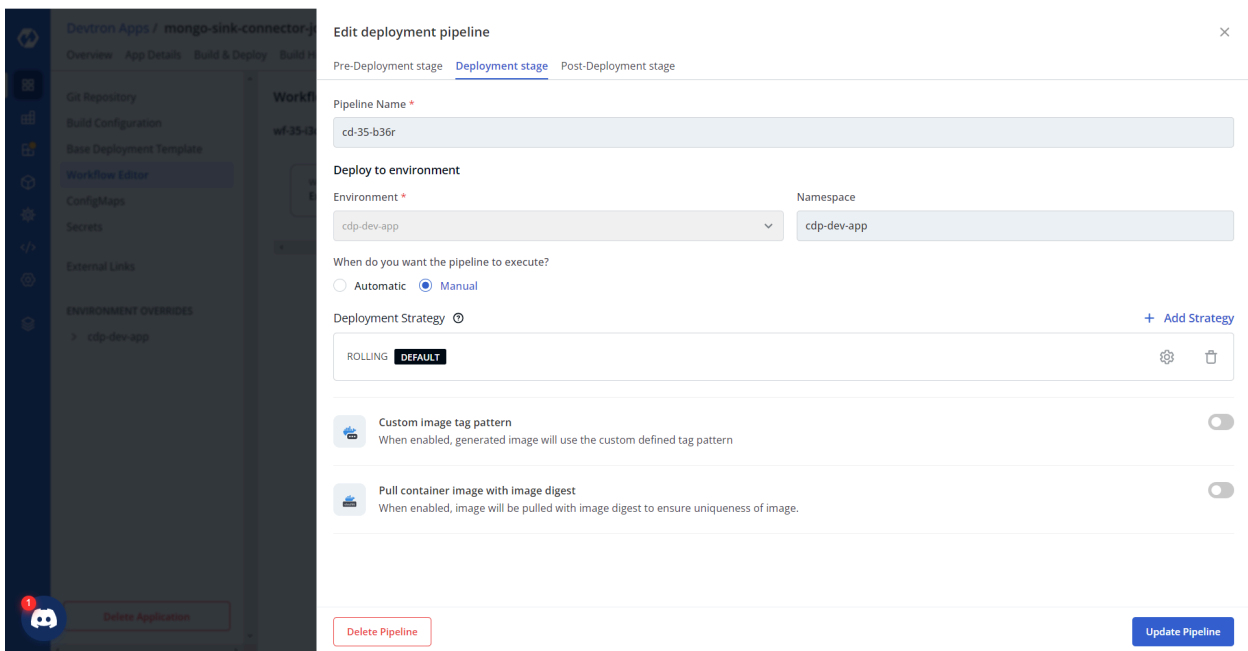
6. Now, click **Try it out** tab below the Auto-generate token section, and update api-token and dockerImage details.

- Click **Execute**, and on successful execution, check the server response.

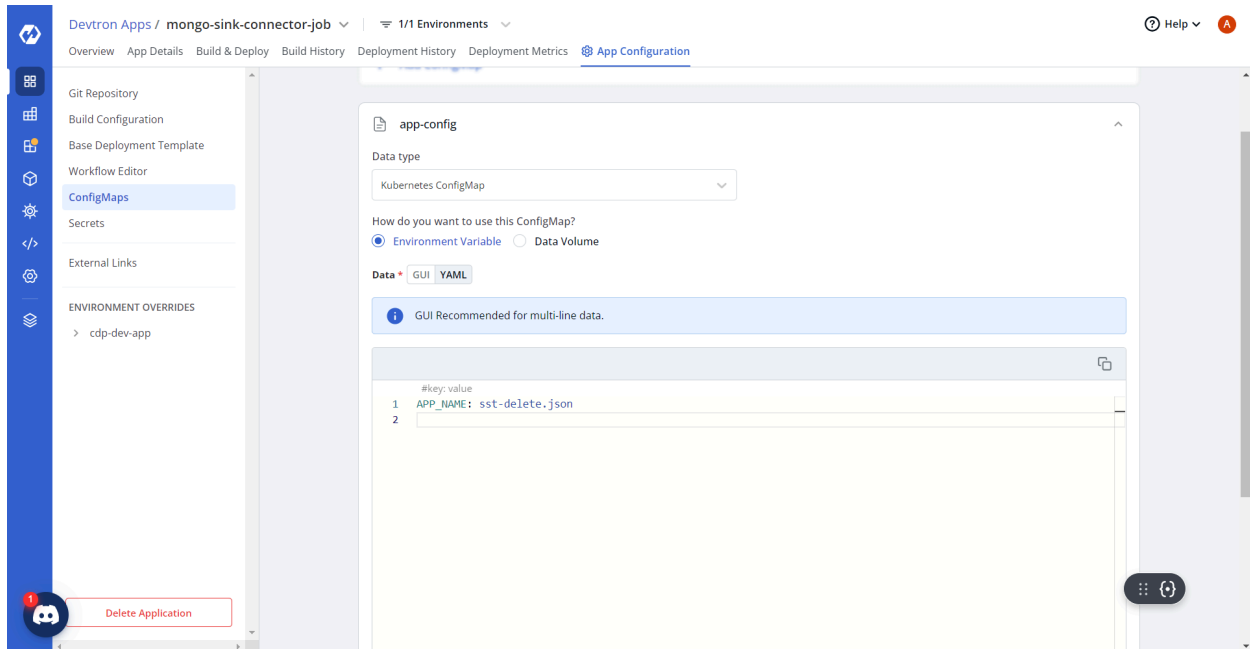


Configuring and deploying custom app

- Click Edit deployment pipeline, check the deployment stage of the workflow and click **Update Pipeline**.



- In the ConfigMaps section, under the **App Configuration** tab, update the app-config as shown below.



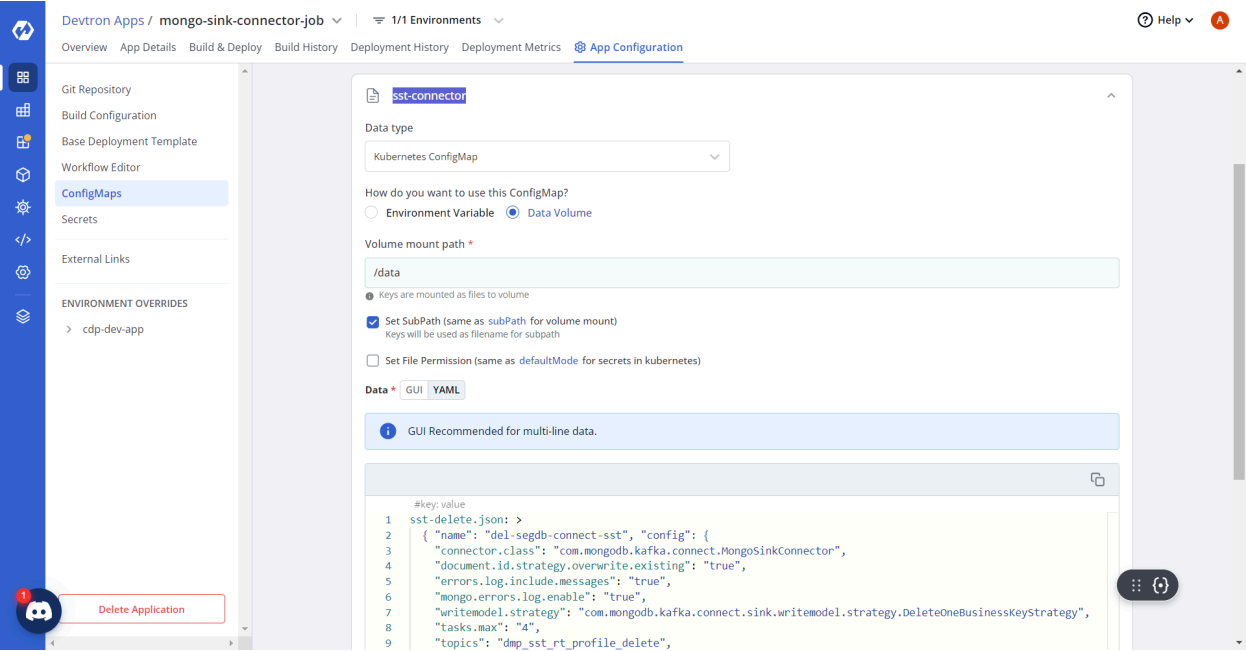
3. Similarly, update ConfigMaps for sst-connector as shown below.

```
sst-delete.json: >
{ "name": "del-segdb-connect-sst", "config": {
  "connector.class": "com.mongodb.kafka.connect.MongoSinkConnector",
  "document.id.strategy.override.existing": "true",
  "errors.log.include.messages": "true",
  "mongo.errors.log.enable": "true",
  "writemodel.strategy":
"com.mongodb.kafka.connect.sink.writemodel.strategy.DeleteOneBusinessKeyStrategy",
  "tasks.max": "4",
  "topics": "dmp_sst_rt_profile_delete",
  "namespace.mapper.value.collection.field": "campaignId",
  "collection": "profile",
  "namespace.mapper": "com.mongodb.kafka.connect.sink.namespace.mapping.FieldPathNamespaceMapper",
  "errors.deadletterqueue.context.headers.enable": "true",
  "mongo.errors.tolerance": "all",
  "database": "segmentation",
  "document.id.strategy": "com.mongodb.kafka.connect.sink.processor.id.strategy.PartialValueStrategy",
  "document.id.strategy.partial.value.projection.list": "key",
  "namespace.mapper.error.if.invalid": "false",
  "errors.deadletterqueue.topic.name": "mongo_profile_delete_dlq",
  "connection.uri":
"mongodb://segdbconnector:46WNWhr6qXgm2JRmTdlyA==@puapso1ahxcdmgodbs1001.hxcd.aws.hclsw.internal:51090,
puapso1bhxcdmgodbs1002.hxcd.aws.hclsw.internal:51090,puapso1ahxcdmgodbs1003.hxcd.aws.hclsw.internal:5109
0,puapso1bhxcdmgodbs1004.hxcd.aws.hclsw.internal:51090/?replicaSet=cdprsg&authSource=admin",
  "errors.tolerance": "all",
  "name": "del-segdb-connect-sst",
  "value.converter": "org.apache.kafka.connect.storage.StringConverter",
  "document.id.strategy.partial.value.projection.type": "AllowList",
  "post.processor.chain": "com.mongodb.kafka.connect.sink.processor.DocumentIdAdder",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "transforms": "customTransformer",
```

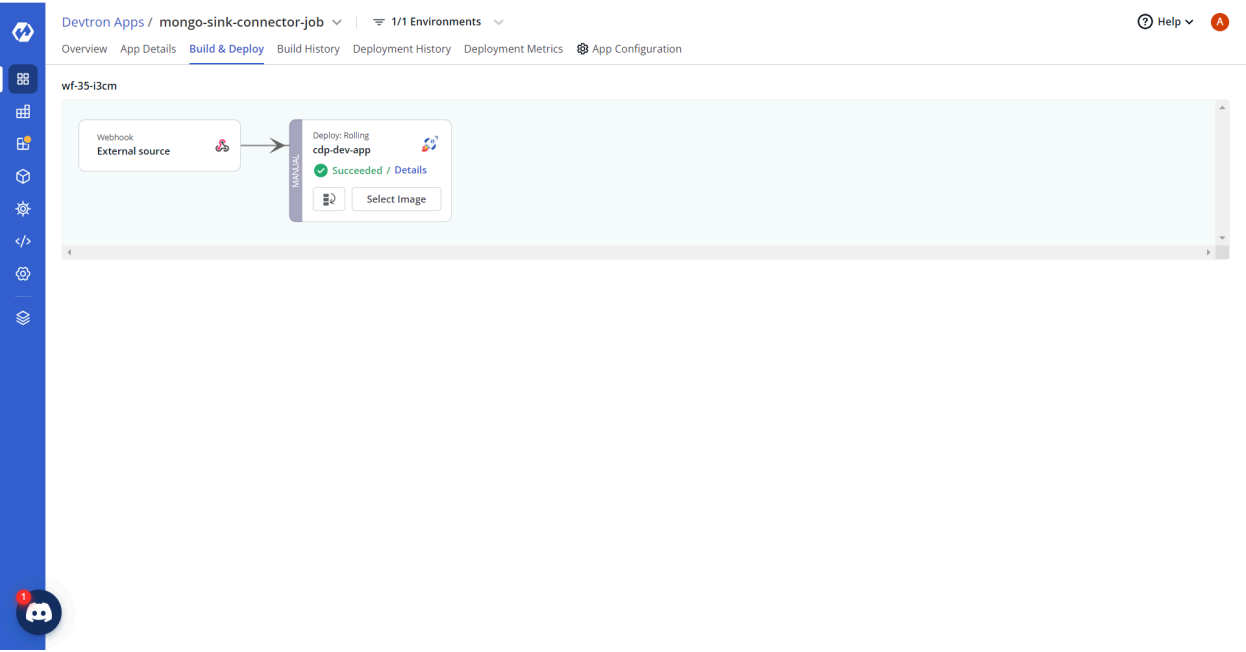
```

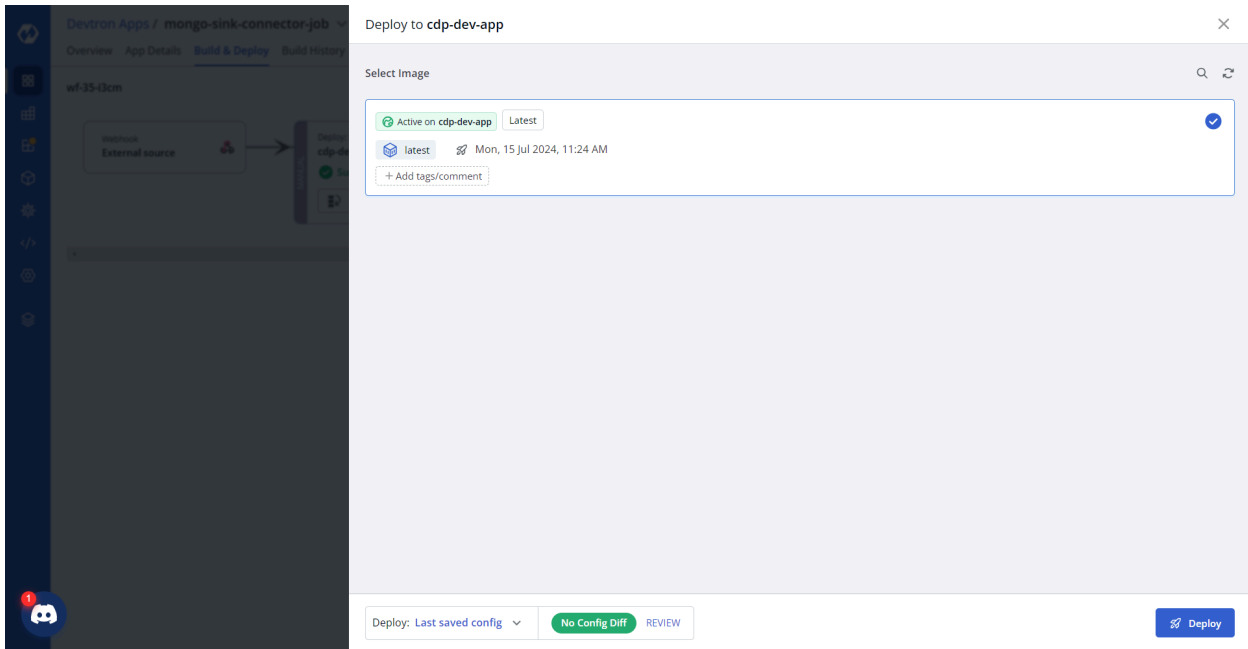
    "transforms.customTransformer.type":
    "co.lemnisk.kafka.connect.transforms.MongoSinkTransformer$Value"
  }
},
sst-update.json: >
{ "name": "kafka-segdb-connect-sst", "config": {
  "connector.class": "com.mongodb.kafka.connect.MongoSinkConnector",
  "errors.log.include.messages": "true",
  "mongo.errors.log.enable": "true",
  "writemodel.strategy":
  "com.mongodb.kafka.connect.sink.writemodel.strategy.ReplaceOneBusinessKeyStrategy",
  "tasks.max": "12",
  "namespace.mapper.value.collection.field": "campaignId",
  "transforms": "customTransformer,tsConverter",
  "transforms.customTransformer.type":
  "co.lemnisk.kafka.connect.transforms.MongoSinkTransformer$Value",
  "transforms.tsConverter.type": "org.apache.kafka.connect.transforms.TimestampConverter$Value",
  "transforms.tsConverter.field": "expts",
  "transforms.tsConverter.target.type": "Date",
  "transforms.tsConverter.format": "yyyy-MM-dd",
  "errors.deadletterqueue.context.headers.enable": "true",
  "mongo.errors.tolerance": "all",
  "database": "segmentation",
  "document.id.strategy": "com.mongodb.kafka.connect.sink.processor.id.strategy.PartialValueStrategy",
  "namespace.mapper.error.if.invalid": "false",
  "document.id.strategy.partial.value.projection.type": "AllowList",
  "value.converter": "org.apache.kafka.connect.storage.StringConverter",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "document.id.strategy.overwrite.existing": "true",
  "topics": "dmp_sst_profile",
  "collection": "profile",
  "namespace.mapper": "com.mongodb.kafka.connect.sink.namespace.mapping.FieldPathNamespaceMapper",
  "document.id.strategy.partial.value.projection.list": "key_type,key",
  "errors.deadletterqueue.topic.name": "mongo_profile_upsert_dlq",
  "connection.uri":
  "mongodb://segdbconnector:46WNWhr6qXgm2JRmTdlyA==@puapso1ahxcdmgodbs1001.hxcd.aws.hclsw.internal:51090,
  puapso1bhxcdmgodbs1002.hxcd.aws.hclsw.internal:51090,puapso1ahxcdmgodbs1003.hxcd.aws.hclsw.internal:5109
  0,puapso1bhxcdmgodbs1004.hxcd.aws.hclsw.internal:51090/?replicaSet=cdprsg&authSource=admin",
  "name": "kafka-segdb-connect-sst",
  "value.converter.schemas.enable": "false",
  "errors.tolerance": "all",
  "post.processor.chain": "com.mongodb.kafka.connect.sink.processor.DocumentIdAdder"
  }
}

```

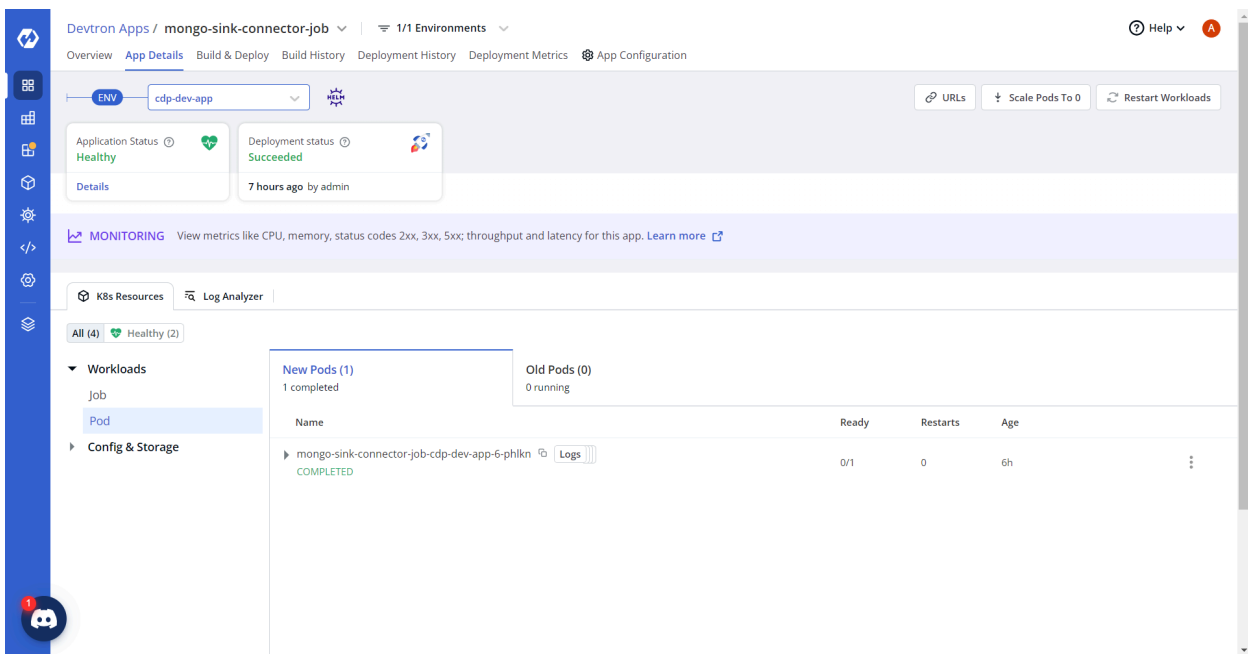


4. Save the configuration, navigate to the **Build & Deploy** tab, and run the execution.





5. On successful deployment, the Pod is in completed status as shown below.

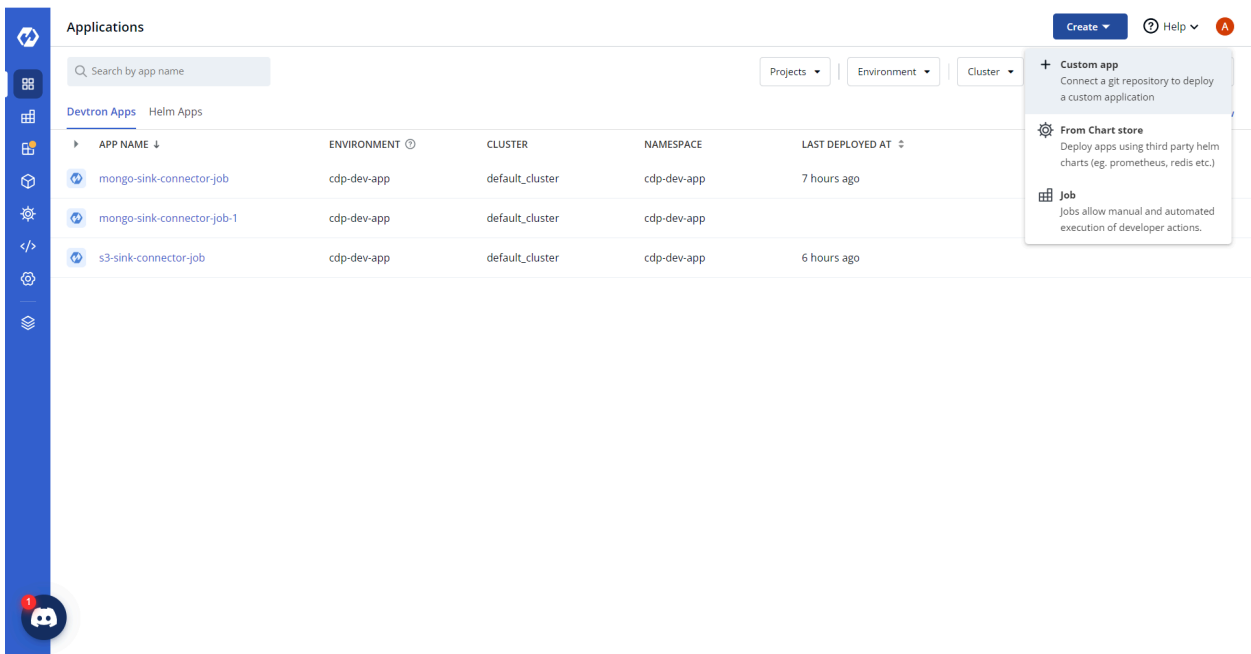


Deploying CDP S3 Connector Sink Job

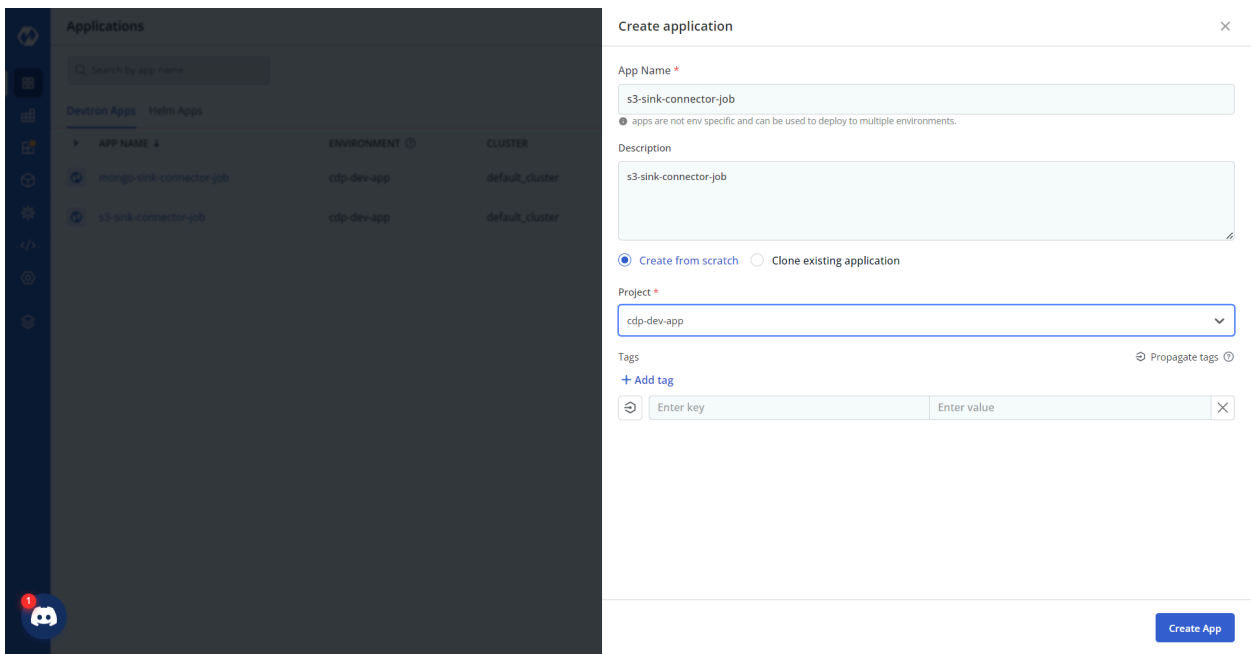
This section provides detailed instructions on how to deploy HCL CDP S3 Connector Sink Job using the Devtron in the OpenShift.

Creating custom app

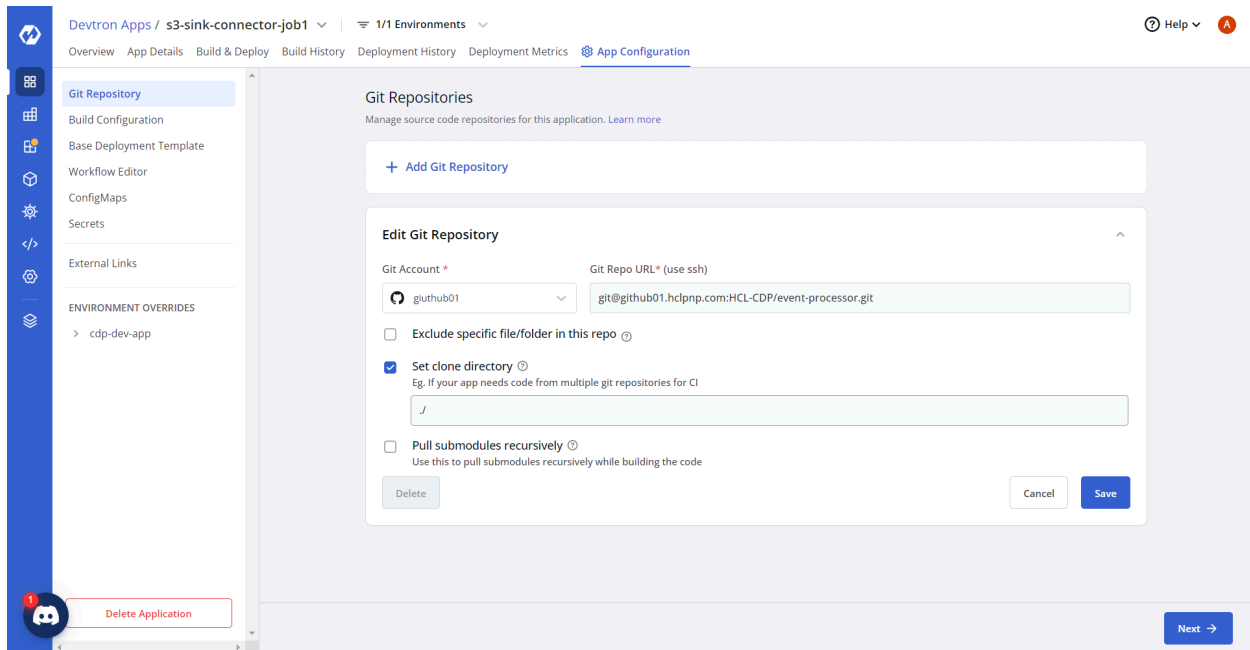
1. In the devtron console, select Applications, and click **Create > Custom app**.



2. In the **Create Application** section, enter application name as "s3-sink-connector-job" and select a project to store the application.



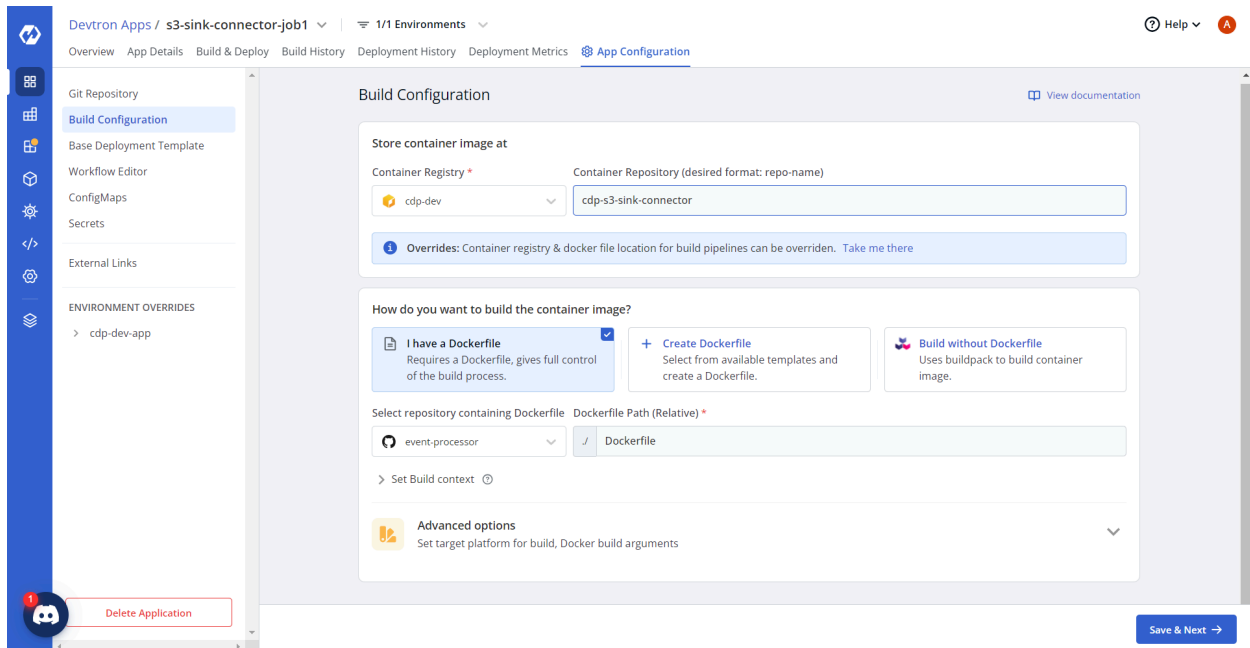
3. Click **Create App** to create the application. Now, click **App Configuration** tab, and update source code repositories for the application.



Building custom app

After successfully creating the custom app, build the custom app with docker image, and configure a workflow for the custom app.

1. In the Devtron Apps page, click **Build Configurations**. In the **App Configuration** tab, select **I have a Dockerfile** option and update the dockerfile path.



2. Click **Save & Next**, and in the **Base Deployment Template** section, update chart type and Mongo Connector Sink URLs in the base deployment template as shown below.

Deployment Template

```
ContainerPort:
  - envoyPort: 8799
    idleTimeout: 1800s
    name: app
    port: 8080
    servicePort: 80
    supportStreaming: true
    useHTTP2: true
EnvVariables: []
GracePeriod: 30
MaxSurge: 1
MaxUnavailable: 0
MinReadySeconds: 60
Spec:
  Affinity:
    Key: null
    Values: nodes
    key: ""
  args:
    enabled: true
    value:
      - /bin/sh
      - -c
      - >
        until $(curl --output /dev/null --silent --head --fail
        http://cdp-s3-sink-connector-service.cdp-dev-app.svc.cluster.local:80); do
```



```

    echo "Waiting for Kafka Connect API..."
    sleep 5
done
    echo "Posting Kafka Connect configuration..."
    curl -X POST -H "Content-Type: application/json" --data @/data/${APP_NAME}
http://cdp-s3-sink-connector-service.cdp-dev-app.svc.cluster.local:80/connectors
command:
  enabled: false
  value: []
containerSecurityContext:
  allowPrivilegeEscalation: false
containers: []
cronjobConfigs:
  concurrencyPolicy: Allow
  failedJobsHistoryLimit: 1
  restartPolicy: OnFailure
  schedule: "* * * * *"
  startingDeadlineSeconds: 100
  successfulJobsHistoryLimit: 3
  suspend: false
ephemeralContainers: []
image:
  pullPolicy: IfNotPresent
imagePullSecrets: []
initContainers: []
jobConfigs:
  activeDeadlineSeconds: 100
  backoffLimit: 5
  completions: 1
  parallelism: 1
  suspend: false
kedaAutoscaling:
  envSourceContainerName: ""
  failedJobsHistoryLimit: 5
  maxReplicaCount: 2
  minReplicaCount: 1
  pollingInterval: 30
  rolloutStrategy: default
  scalingStrategy:
    customScalingQueueLengthDeduction: 1
    customScalingRunningJobPercentage: "0.5"
    multipleScalersCalculation: max
    pendingPodConditions:
      - Ready
      - PodScheduled
      - AnyOtherCustomPodCondition
    strategy: custom
  successfulJobsHistoryLimit: 5
triggerAuthentication:
  enabled: false
  name: ""
  spec: {}
triggers:
- authenticationRef: {}
  metadata:
    host: RabbitMqHost
    queueLength: "5"
    queueName: hello

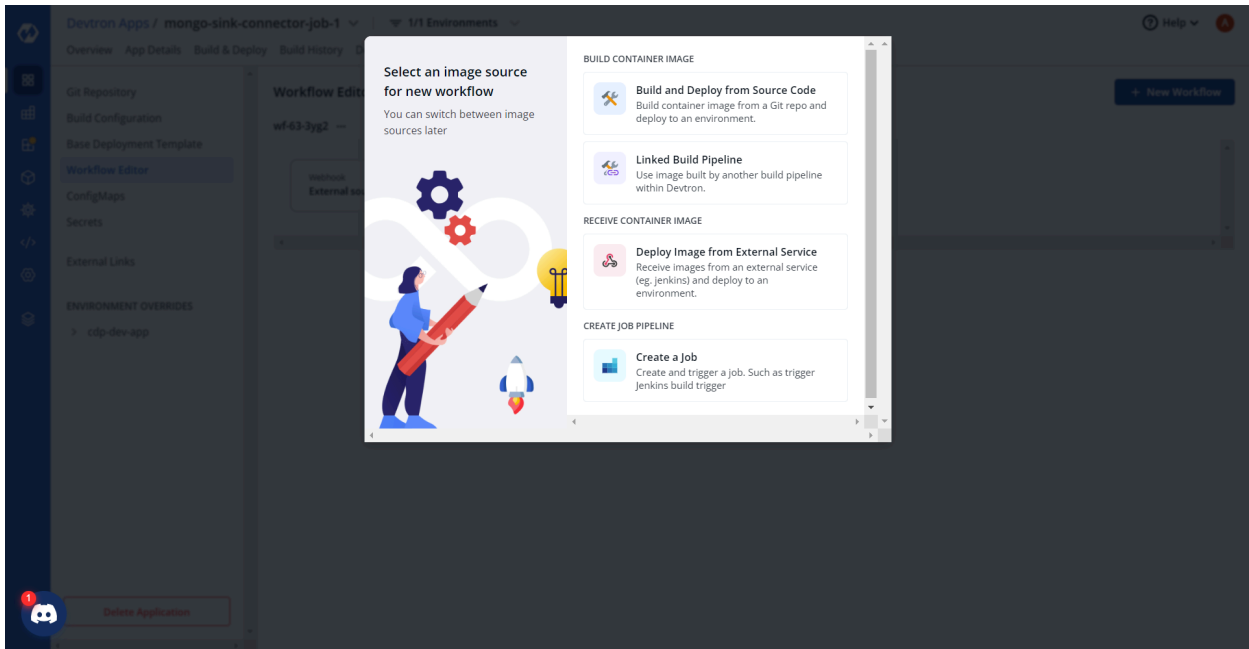
```

```

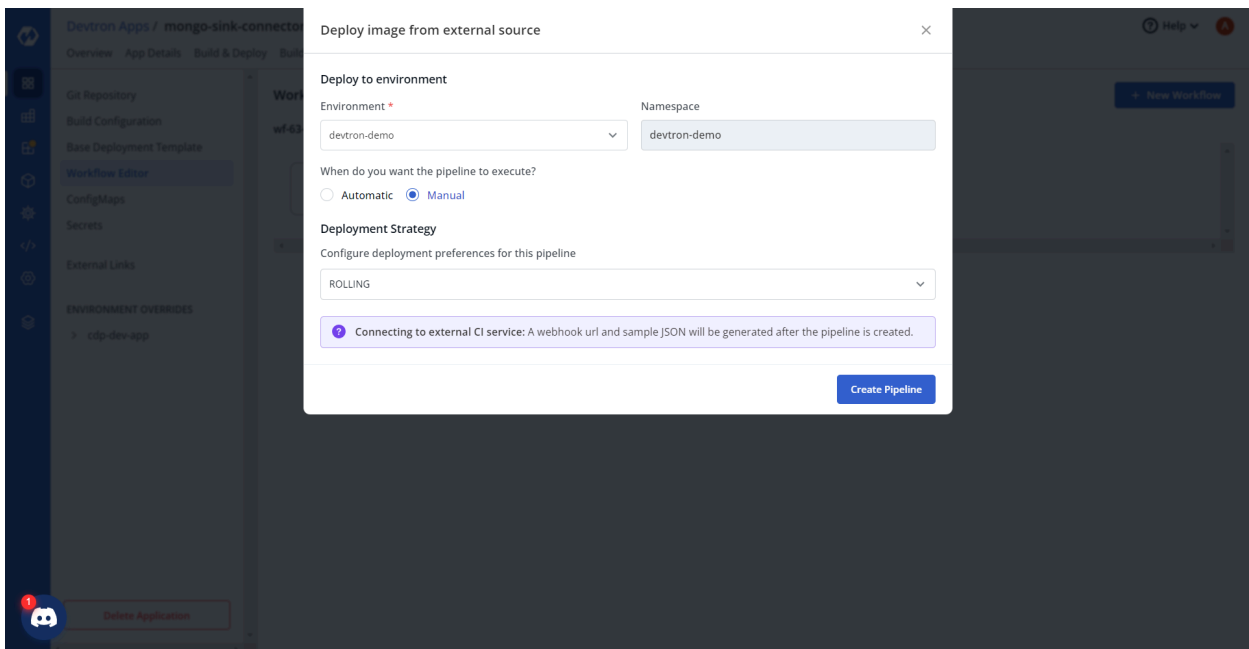
    type: rabbitmq
kind: Job
pauseForSecondsBeforeSwitchActive: 30
podAnnotations: {}
podLabels: {}
podSecurityContext: {}
prometheus:
  release: monitoring
rawYaml: []
readinessGates: []
resources:
  limits:
    cpu: "0.05"
    memory: 50Mi
  requests:
    cpu: "0.01"
    memory: 10Mi
secret:
  data: {}
  enabled: false
server:
  deployment:
    image: ""
    image_tag: 1-95af053
service:
  annotations: {}
  enabled: false
  type: ClusterIP
servicemonitor:
  additionalLabels: {}
setHostnameAsFQDN: false
shareProcessNamespace: false
tolerations: []
topologySpreadConstraints: []
volumeMounts: []
volumes: []
waitForSecondsBeforeScalingDown: 30

```

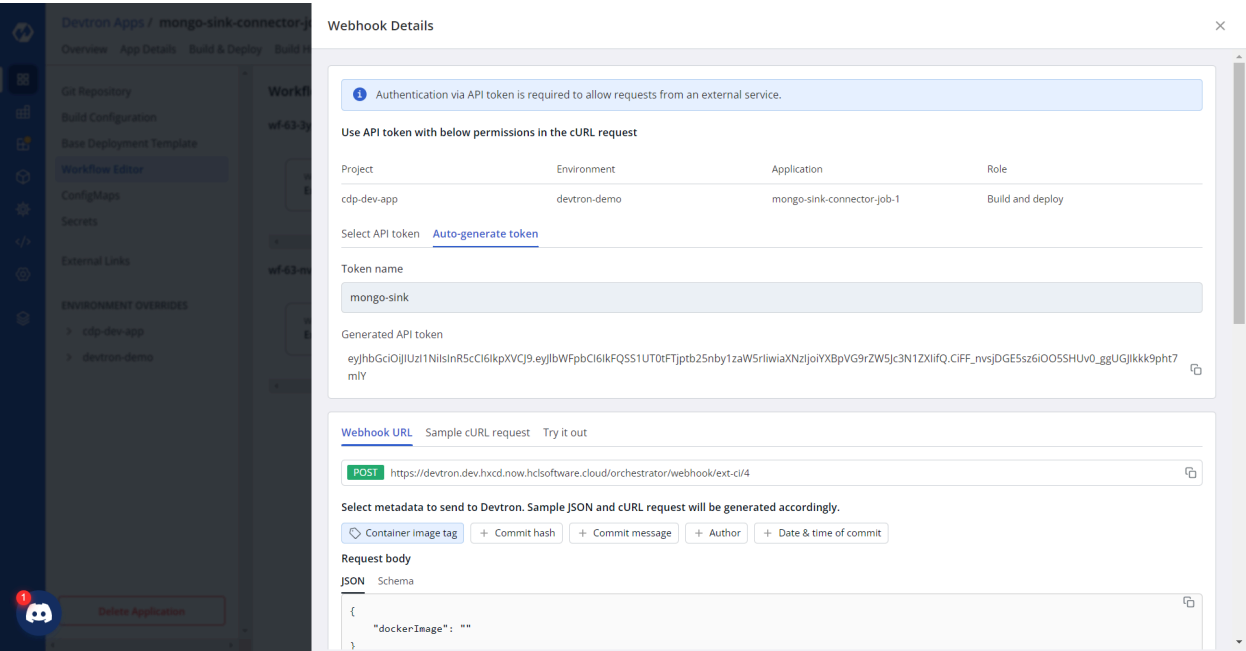
3. Click **Save & Next**. In the Workflow Editor section, add new workflow, and select the **Deploy Image from External Source** option.



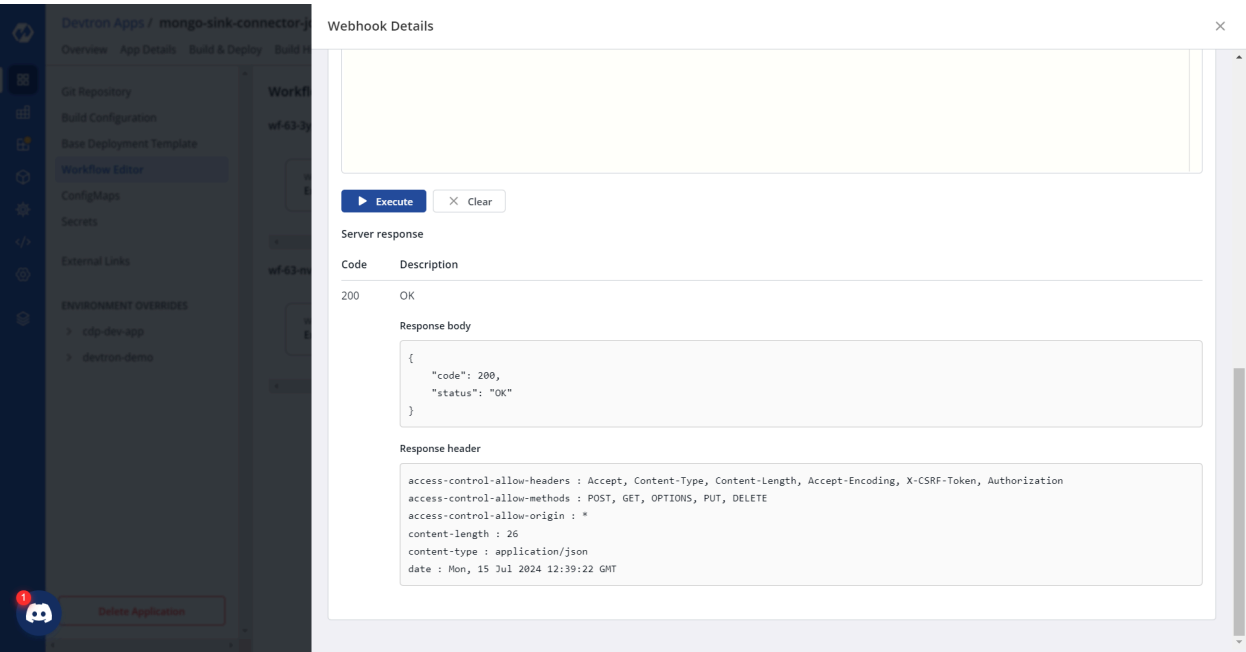
4. Configure required parameters, and click **Create pipeline** at the right bottom corner of the Deploy image from external source section.



5. In the Workflow Editor, click **Edit Workflow**, and in the **Webhook Details** section, click the **Auto Generate Token** tab, and copy the token details.

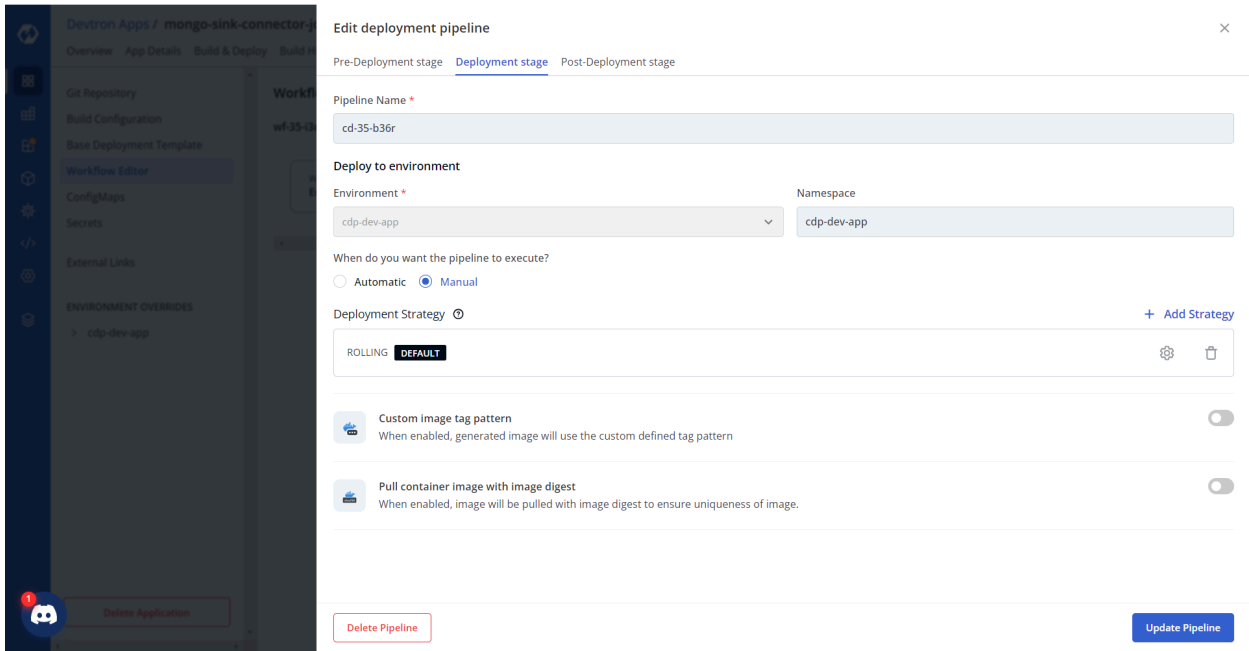


- 6. Now, click **Try it out** tab below the Auto-generate token section, and update api-token and dockerImage details.
- 7. Click **Execute**, and on successful execution, check the server response.

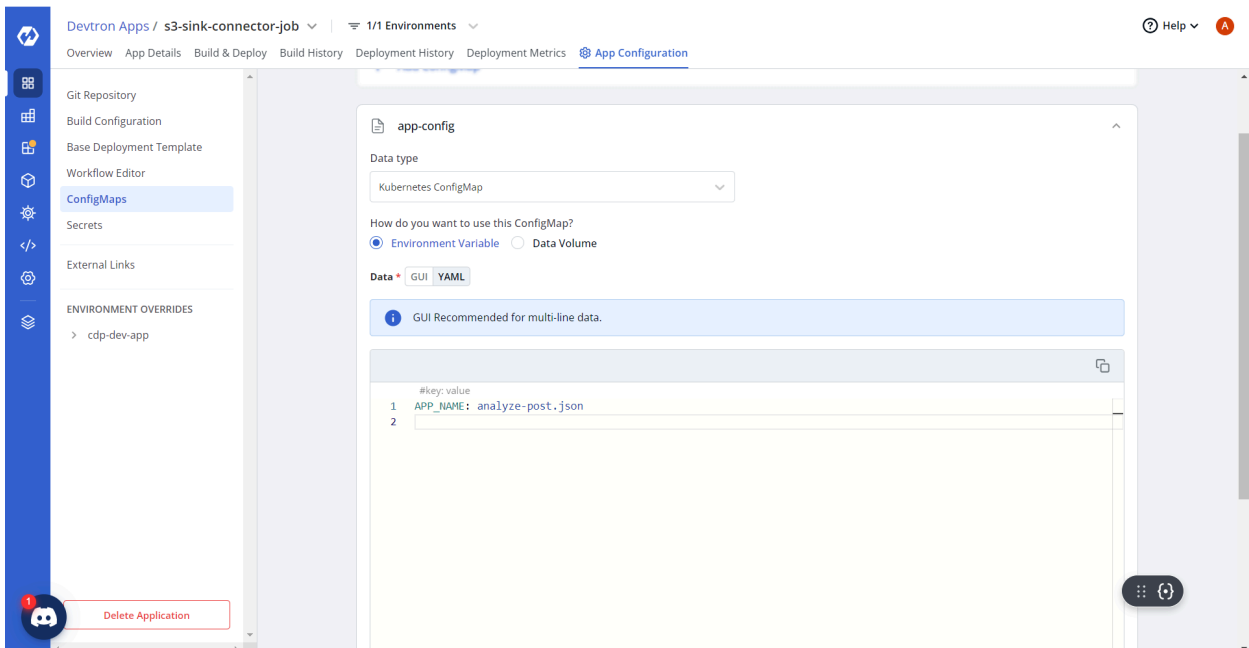


Configuring and deploying custom app

1. Click Edit deployment pipeline, check the deployment stage of the workflow and click **Update Pipeline**.



2. In the **ConfigMaps** section, under the **App Configuration** tab, update the app-config as shown below.



3. Similarly, update the ConfigMaps for sst-connector as shown below.

```
sst-connector.json: >
{
  "name": "s3-sink-connect-sst",
```

```

    "config": {
      "connector.class": "io.confluent.connect.s3.S3SinkConnector",
      "errors.log.include.messages": "true",
      "s3.region": "ap-south-1",
      "topics.dir": "",
      "partition.field.format.path": "false",
      "flush.size": "10000",
      "tasks.max": "6",
      "s3.part.size": "5242880",
      "timezone": "UTC",
      "transforms": "GenericTransformer",
      "rotate.interval.ms": "-1",
      "locale": "US",
      "errors.deadletterqueue.context.headers.enable": "true",
      "format.class": "io.confluent.connect.s3.format.bytearray.ByteArrayFormat",
      "transforms.GenericTransformer.type":
"co.lemnisk.kafka.connect.transforms.GenericTransformer$Value",
      "aws.access.key.id": "",
      "errors.deadletterqueue.topic.replication.factor": "2",
      "value.converter": "org.apache.kafka.connect.json.JsonConverter",
      "errors.log.enable": "true",
      "s3.bucket.name": "hclsw-hxcd-hss-prd-s3-cdp-app",
      "key.converter": "org.apache.kafka.connect.storage.StringConverter",
      "partition.duration.ms": "300000",

      "topics": "sst_blacklistProfile,sst_mergeProfile,sst_outboundEvents,sst_inboundEvents,sst_failedEvents,s
st_blacklist,sst_deleted0id,sst_profile","directory.delim": "/", "aws.secret.access.key": "", "transforms.Ge
nericTransformer.meta": "[{\"topic\": \"sst_blacklistProfile\", \"tsCol\": \"ts\", \"dataFormat\":
\"json\"}, {\"topic\": \"sst_mergeProfile\", \"tsCol\": \"ts\", \"dataFormat\": \"json\"}, {\"topic\":
\"sst_outboundEvents\", \"tsCol\": \"ts\", \"dataFormat\": \"json\"}, {\"topic\": \"sst_inboundEvents\",
\"tsCol\": \"ts\", \"dataFormat\": \"json\"}, {\"topic\": \"sst_failedEvents\", \"tsCol\": \"ts\",
\"dataFormat\": \"json\"}, {\"topic\": \"sst_blacklist\", \"tsCol\": \"4\", \"dataFormat\":
\"tsv\"}, {\"topic\": \"sst_deleted0id\", \"tsCol\": \"5\", \"dataFormat\": \"tsv\"}, {\"topic\":
\"sst_profile\", \"tsCol\": \"ts\", \"dataFormat\": \"json\"}]",
      "partition.field.name": "p1,p2",
      "s3.compression.type": "gzip",
      "errors.deadletterqueue.topic.name": "dlq_sst",
      "partitioner.class": "co.lemnisk.kafka.connect.partitionner.GenericPartitioner",
      "value.converter.schemas.enable": "false",
      "name": "s3-sink-connect-sst",
      "errors.tolerance": "all",
      "storage.class": "io.confluent.connect.s3.storage.S3Storage",
      "rotate.schedule.interval.ms": "900000",
      "path.format": "YYYY/MM/dd",
      "timestamp.extractor": "Record"
    }
  },
diapi-connector.json: >
{
  "name": "s3-sink-connect-diapi",
  "config": {
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "errors.log.include.messages": "true",
    "s3.region": "ap-south-1",
    "topics.dir": "",
    "partition.field.format.path": "false",
    "flush.size": "10000",
    "tasks.max": "6",

```

```

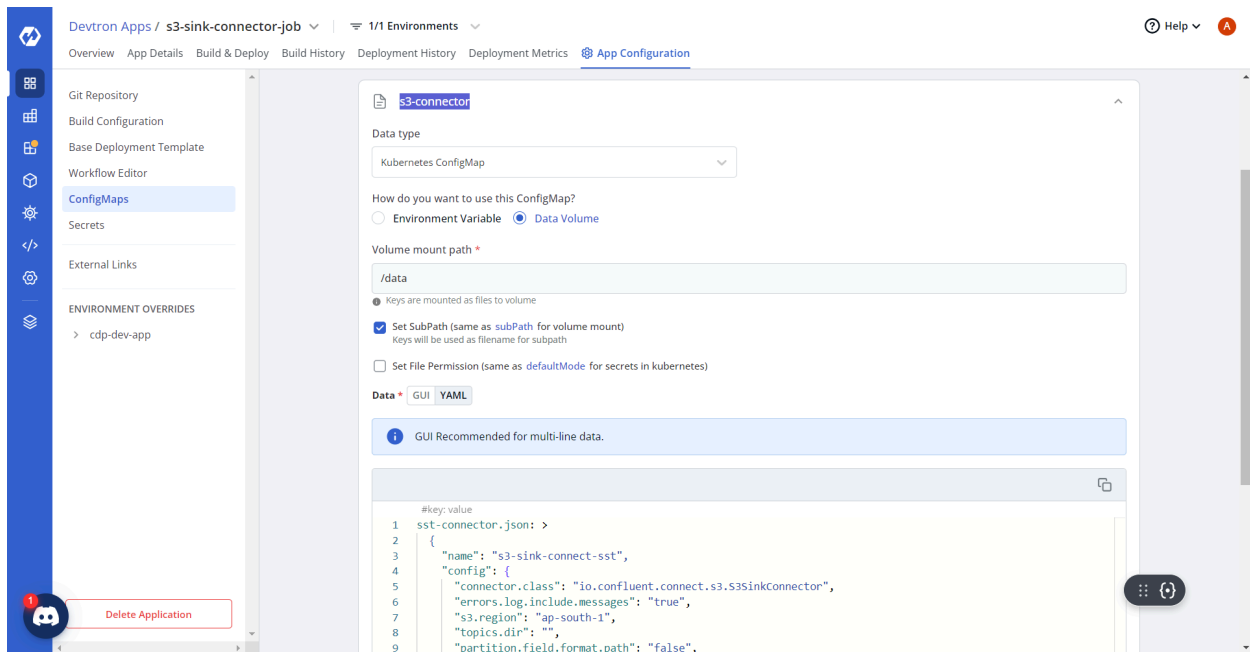
    "s3.part.size": "5242880",
    "timezone": "UTC",
    "transforms": "GenericTransformer",
    "rotate.interval.ms": "-1",
    "locale": "US",
    "errors.deadletterqueue.context.headers.enable": "true",
    "format.class": "io.confluent.connect.s3.format.bytearray.ByteArrayFormat",
    "transforms.GenericTransformer.type":
"co.lemnisk.kafka.connect.transforms.GenericTransformer$Value",
    "aws.access.key.id": "",
    "errors.deadletterqueue.topic.replication.factor": "2",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "errors.log.enable": "true",
    "s3.bucket.name": "hclsw-hxcd-hss-prd-s3-cdp-app",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "partition.duration.ms": "300000",
    "topics": "diapi_outbound",
    "directory.delim": "/",
    "aws.secret.access.key": "",
    "transforms.GenericTransformer.meta":
"[{\\"topic\\":\\"diapi_outbound\\",\\"tsFormat\\":\\"yyyy-MM-dd'T'HH:mm:ss\\",\\"tsCol\\":\\"ts\\",\\"dataFormat\\":
\\"json\\",\\"inputTz\\":\\"UTC\\",\\"outputTz\\":\\"Asia/Kolkata\\"}]",
    "partition.field.name": "eventType,campaignId",
    "s3.compression.type": "gzip",
    "errors.deadletterqueue.topic.name": "dlq_diapi",
    "partitioner.class": "co.lemnisk.kafka.connect.partitionner.GenericPartitioner",
    "value.converter.schemas.enable": "false",
    "name": "s3-sink-connect-diapi",
    "errors.tolerance": "all",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "rotate.schedule.interval.ms": "900000",
    "path.format": "YYYY/MM/dd",
    "timestamp.extractor": "Record"
  }
},
analyze-post.json: >
{ "name": "s3-sink-connect-analyze_post", "config": {
  "connector.class": "io.confluent.connect.s3.S3SinkConnector",
  "errors.log.include.messages": "true",
  "s3.region": "ap-south-1",
  "topics.dir": "",
  "partition.field.format.path": "false",
  "flush.size": "10000",
  "tasks.max": "6",
  "s3.part.size": "5242880",
  "timezone": "UTC",
  "transforms": "GenericTransformer",
  "rotate.interval.ms": "-1",
  "locale": "US",
  "errors.deadletterqueue.context.headers.enable": "true",
  "format.class": "io.confluent.connect.s3.format.bytearray.ByteArrayFormat",
  "transforms.GenericTransformer.type":
"co.lemnisk.kafka.connect.transforms.GenericTransformer$Value",
  "aws.access.key.id": "",
  "errors.deadletterqueue.topic.replication.factor": "2",
  "value.converter": "org.apache.kafka.connect.storage.StringConverter",
  "errors.log.enable": "true",
  "s3.bucket.name": "hclsw-hxcd-hss-prd-s3-cdp-app",

```

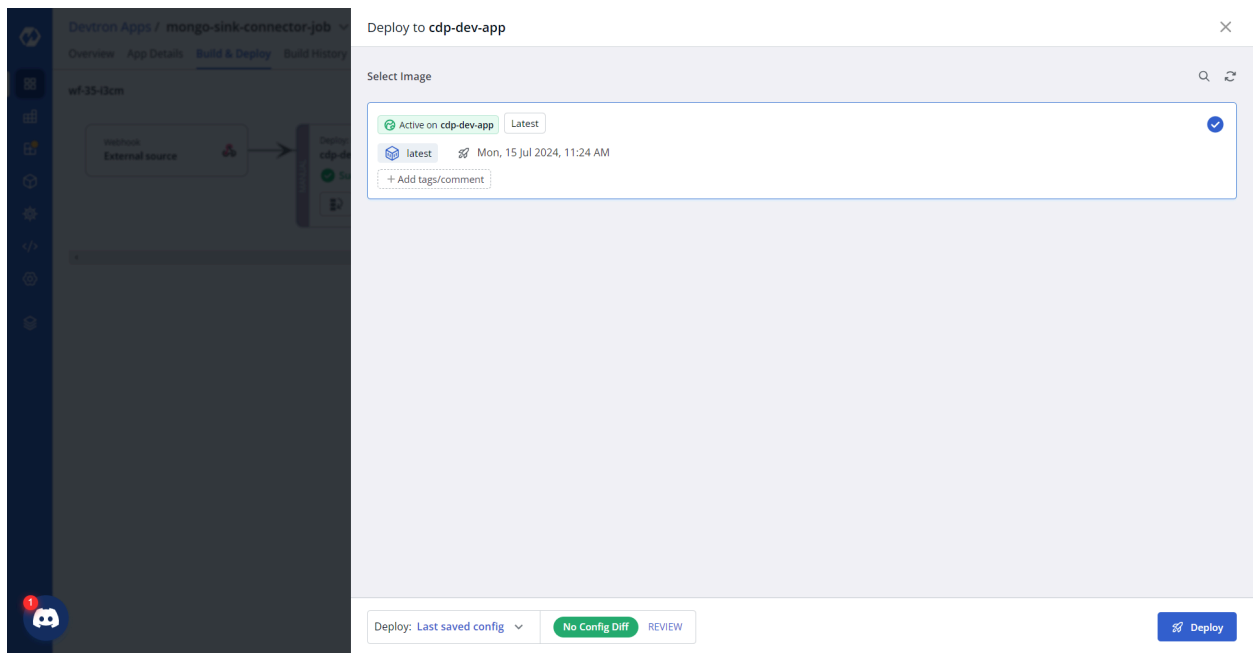
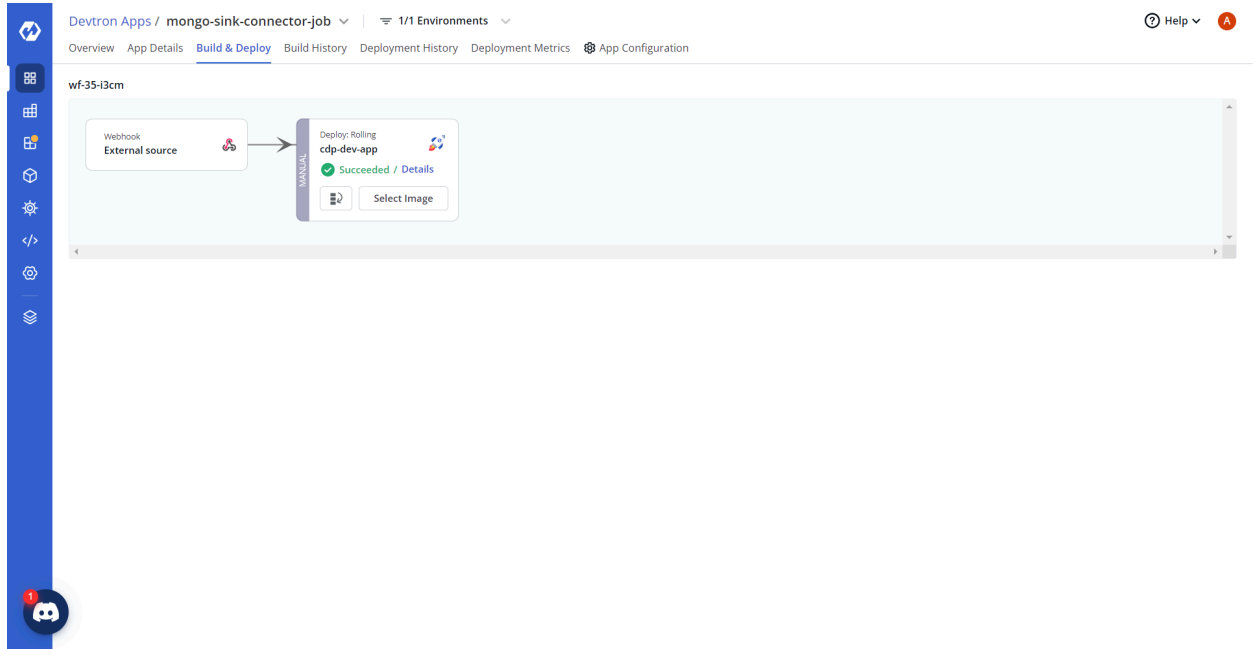
```

    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "partition.duration.ms": "300000",
    "topics": "analyze_post",
    "directory.delim": "/",
    "aws.secret.access.key": "",
    "transforms.GenericTransformer.meta":
    "[{"topic":"analyze_post","tsFormat":"yyyy-MM-dd'T'HH:mm:ss","dataFormat":"json",
    \ "inputFormat":"tsv","dataCol":3,"partitionerKeys":"AnalyzePost,{campaignId}","tsCol":
    \ "receivedAt","inputTz":"UTC","outputTz":"Asia/Kolkata"}]",
    "s3.compression.type": "gzip",
    "errors.deadletterqueue.topic.name": "dlq_analyze_post",
    "partitioner.class": "co.lemnisk.kafka.connect.partitioners.GenericPartitioner",
    "value.converter.schemas.enable": "false",
    "name": "s3-sink-connect-analyze_post",
    "errors.tolerance": "all",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "rotate.schedule.interval.ms": "900000",
    "path.format": "YYYY/MM/dd",
    "timestamp.extractor": "Record"
  } }

```



4. Save the configuration, navigate to the **Build & Deploy** tab, and run the execution.



5. On successful deployment, the Pod is in completed status as shown below.

