Red Hat OpenShift Pipelines 1.13

Custom Tekton Hub instance

Installing a custom instance of Tekton Hub

Last Updated: 2023-12-08

Red Hat OpenShift Pipelines 1.13 Custom Tekton Hub instance

Installing a custom instance of Tekton Hub

Legal Notice

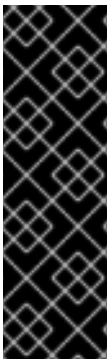
Abstract

This document provides information about installing and deploying a custom instance of Tekton Hub.

Table of Contents

CHAPTER 1. USING TEKTON HUB WITH OPENSHIFT PIPELINES 3
1.1. INSTALLING AND DEPLOYING TEKTON HUB ON A OPENSHIFT CONTAINER PLATFORM CLUSTER 3
1.1.1. Installing Tekton Hub without login and rating 3
1.1.2. Installing Tekton Hub with login and rating 5
1.2. OPTIONAL: USING A CUSTOM DATABASE IN TEKTON HUB 8
1.2.1. Optional: Installing Crunchy Postgres database and Tekton Hub 9
1.2.2. Optional: Migrating Tekton Hub data to an existing Crunchy Postgres database 12
1.3. UPDATING TEKTON HUB WITH CUSTOM CATEGORIES AND CATALOGS 15
1.4. MODIFYING THE CATALOG REFRESH INTERVAL OF TEKTON HUB 16
1.5. ADDING NEW USERS IN TEKTON HUB CONFIGURATION 17
1.6. DISABLING TEKTON HUB AUTHORIZATION AFTER UPGRADING THE RED HAT OPENSHIFT PIPELINES OPERATOR FROM 1.7 TO 1.8 18
1.7. ADDITIONAL RESOURCES 19

CHAPTER 1. USING TEKTON HUB WITH OPENSHIFT PIPELINES



IMPORTANT

Tekton Hub is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

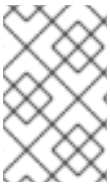
Tekton Hub helps you discover, search, and share reusable tasks and pipelines for your CI/CD workflows. A public instance of Tekton Hub is available at hub.tekton.dev. Cluster administrators can also install and deploy a custom instance of Tekton Hub by modifying the configurations in the **TektonHub** custom resource (CR).

1.1. INSTALLING AND DEPLOYING TEKTON HUB ON A OPENSHIFT CONTAINER PLATFORM CLUSTER

Tekton Hub is an optional component; cluster administrators cannot install it using the **TektonConfig** custom resource (CR). To install and manage Tekton Hub, use the **TektonHub** CR.

You can install Tekton Hub on your cluster using two modes:

- *Without* login authorization and ratings for Tekton Hub artifacts
- *with* login authorization and ratings for Tekton Hub artifacts



NOTE

If you are using Github Enterprise or Gitlab Enterprise, install and deploy Tekton Hub in the same network as the enterprise server. For example, if the enterprise server is running behind a VPN, deploy Tekton Hub on a cluster that is also behind the VPN.

1.1.1. Installing Tekton Hub without login and rating

You can install Tekton Hub on your cluster automatically with default configuration. When using the default configuration, Tekton Hub does not support login with authorization and ratings for Tekton Hub artifacts.

Prerequisites

- Ensure that the Red Hat OpenShift Pipelines Operator is installed in the default **openshift-pipelines** namespace on the cluster.

Procedure

1. Create a **TektonHub** CR similar to the following example.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines (1)
  db:                  # Optional: If you want to use custom database
    secret: tekton-hub-db  # Name of db secret should be `tekton-hub-db`

  categories:            # Optional: If you want to use custom categories
    - Automation
    - Build Tools
    - CLI
    - Cloud
    - Code Quality
    - ...

  catalogs:              # Optional: If you want to use custom catalogs
    - name: tekton
      org: tektoncd
      type: community
      provider: github
      url: https://github.com/tektoncd/catalog
      revision: main

  scopes:                # Optional: If you want to add new users
    - name: agent:create
      users: [abc, qwe, pqr]
    - name: catalog:refresh
      users: [abc, qwe, pqr]
    - name: config:refresh
      users: [abc, qwe, pqr]

  default:               # Optional: If you want to add custom default scopes
    scopes:
      - rating:read
      - rating:write

  api:
    catalogRefreshInterval: 30m (2)
```

(1) The namespace in which Tekton Hub must be installed; default is **openshift-pipelines**.

(2) The time interval after which the catalog refreshes automatically. The supported units of time are seconds (**s**), minutes (**m**), hours (**h**), days (**d**), and weeks (**w**). The default interval is 30 minutes.

NOTE

If you don't provide custom values for the optional fields in the **TektonHub** CR, the default values configured in the Tekton Hub API config map is used.

2. Apply the **TektonHub** CR.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

3. Check the status of the installation. The **TektonHub** CR might take some time to attain steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

1.1.2. Installing Tekton Hub with login and rating

You can install Tekton Hub on your cluster with custom configuration that supports login with authorization and ratings for Tekton Hub artifacts.

Prerequisites

- Ensure that the Red Hat OpenShift Pipelines Operator is installed in the default **openshift-pipelines** namespace on the cluster.

Procedure

1. Create an OAuth application with your Git repository hosting provider, and note the Client ID and Client Secret. The supported providers are GitHub, GitLab, and BitBucket.
   - For a GitHub OAuth application, set the Homepage URL and the Authorization callback URL as **<auth-route>**.
   - For a GitLab OAuth application, set the **REDIRECT_URI** as **<auth-route>/auth/gitlab/callback**.
   - For a BitBucket OAuth application, set the **Callback URL** as **<auth-route>**.
2. Edit the **<tekton_hub_root>/config/02-api/20-api-secret.yaml** file to include the Tekton Hub API secrets. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: tekton-hub-api
  namespace: openshift-pipelines
type: Opaque
stringData:
  GH_CLIENT_ID: 1
  GH_CLIENT_SECRET: 2
  GL_CLIENT_ID: 3
  GL_CLIENT_SECRET: 4
  BB_CLIENT_ID: 5
  BB_CLIENT_SECRET: 6
  JWT_SIGNING_KEY: 7
  ACCESS_JWT_EXPIRES_IN: 8
```

```
REFRESH_JWT_EXPIRES_IN: 9
AUTH_BASE_URL: 10
GHE_URL: 11
GLE_URL: 12
```

1. The Client ID from the GitHub OAuth application.
2. The Client Secret from the GitHub OAuth application.
3. The Client ID from the GitLab OAuth application.
4. The Client Secret from the GitLab OAuth application.
5. The Client ID from the BitBucket OAuth application.
6. The Client Secret from the BitBucket OAuth application.
7. A long, random string used to sign the JSON Web Token (JWT) created for users.
8. Add the time limit after which the access token expires. For example, **1m**, where m denotes minutes. The supported units of time are seconds (**s**), minutes (**m**), hours (**h**), days (**d**), and weeks (**w**).
9. Add the time limit after which the refresh token expires. For example, **1m**, where **m** denotes minutes. The supported units of time are seconds (**s**), minutes (**m**), hours (**h**), days (**d**), and weeks (**w**). Ensure that the expiry time set for token refresh is greater than the expiry time set for token access.
10. Route URL for the OAuth application.
11. GitHub Enterprise URL, if you are authenticating using GitHub Enterprise. Do not provide the URL to the catalog as a value for this field.
12. GitLab Enterprise URL, if you are authenticating using GitLab Enterprise. Do not provide the URL to the catalog as a value for this field.

NOTE

You can delete the unused fields for the Git repository hosting service providers that are irrelevant to your deployment.

3. Create a **TektonHub** CR similar to the following example.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines 1
  db: 2
    secret: tekton-hub-db 3

  categories: 4
    - Automation
```

```
    - Build Tools
    - CLI
    - Cloud
    - Code Quality
      ...

  catalogs: 5
    - name: tekton
      org: tektoncd
      type: community
      provider: github
      url: https://github.com/tektoncd/catalog
      revision: main

  scopes: 6
    - name: agent:create
      users: [<username>]
    - name: catalog:refresh
      users: [<username>]
    - name: config:refresh
      users: [<username>]

  default: 7
    scopes:
      - rating:read
      - rating:write

  api:
    catalogRefreshInterval: 30m 8
```

1. The namespace in which Tekton Hub must be installed; default is **openshift-pipelines**.
2. Optional: Custom database, such as a Crunchy Postgres database.
3. The name of the database secret must be **tekton-hub-db**.
4. Optional: Customized categories for tasks and pipelines in Tekton Hub.
5. Optional: Customized catalogs for Tekton Hub.
6. Optional: Additional users. You can metion multiple users, such as **[<username_1>, <username_2>, <username_3>]**.
7. Optional: Customized default scopes.
8. The time interval after which the catalog refreshes automatically. The supported units of time are seconds (**s**), minutes (**m**), hours (**h**), days (**d**), and weeks (**w**). The default interval is 30 minutes.

NOTE

If you don't provide custom values for the optional fields in the **TektonHub** CR, the default values configured in the Tekton Hub API config map is used.

4. Apply the **TektonHub** CR.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

5. Check the status of the installation. The **TektonHub** CR might take some time to attain steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

1.2. OPTIONAL: USING A CUSTOM DATABASE IN TEKTON HUB

Cluster administrators can use a custom database with Tekton Hub, instead of the default PostgreSQL database installed by the Operator. You can associate a custom database at the time of installation, and use it with the **db-migration**, **api**, and **ui** interfaces provided by Tekton Hub. Alternatively, you can associate a custom database with Tekton Hub even after the installation with the default database is complete.

Procedure

1. Create a secret named **tekton-hub-db** in the target namespace with the following keys:
   - **POSTGRES_HOST**
   - **POSTGRES_DB**
   - **POSTGRES_USER**
   - **POSTGRES_PASSWORD**
   - **POSTGRES_PORT**

   Example: Custom database secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: tekton-hub-db
  labels:
    app: tekton-hub-db
type: Opaque
stringData:
  POSTGRES_HOST: <The name of the host of the database>
  POSTGRES_DB: <Name of the database>
  POSTGRES_USER: <username>
  POSTGRES_PASSWORD: <password>
  POSTGRES_PORT: <The port that the database is listening on>
...
```

NOTE

The default target namespace is **openshift-pipelines**.

2. In the **TektonHub** CR, set the value of the database secret attribute to **tekton-hub-db**.

Example: Adding custom database secret

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines
  db:
    secret: tekton-hub-db
  api:
    hubConfigUrl: https://raw.githubusercontent.com/tektoncd/hub/main/config.yaml
    catalogRefreshInterval: 30m
...
```

3. Use the updated **TektonHub** CR to associate the custom database with Tekton Hub.

   a. If you are associating the custom database at the time of installing Tekton Hub on your cluster, apply the updated **TektonHub** CR.

   ```
   $ oc apply -f <tekton-hub-cr>.yaml
   ```

   b. Alternatively, if you are associating the custom database after the installation of Tekton Hub is complete, replace the existing **TektonHub** CR with the updated **TektonHub** CR.

   ```
   $ oc replace -f <tekton-hub-cr>.yaml
   ```

4. Check the status of the installation. The **TektonHub** CR might take some time to attain steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

1.2.1. Optional: Installing Crunchy Postgres database and Tekton Hub

Cluster administrators can install the Crunchy Postgres database and configure Tekton Hub to use it instead of the default database.

Prerequisites

- Install the Crunchy Postgres Operator from the Operator Hub.
- Create a Postgres instance that initiates a Crunchy Postgres database.

Procedure

1. Get into the Crunchy Postgres pod.

   Example: Getting into the test-instance1-m7hh-0 pod

   ```
   $ oc exec -it -n openshift-operators test-instance1-m7hh-0 -- /bin/sh

   Defaulting container name to database.
   Use 'oc describe pod/test-instance1-m7hh-0 -n openshift-operators' to see all of the
   containers in this pod.
   sh-4.4$ psql -U postgres
   psql (14.4)
   Type "help" for help.
   ```

2. Find the **pg_hba.conf** file.

   ```
   postgres=# SHOW hba_file;
           hba_file
   --------------------------
    /pgdata/pg14/pg_hba.conf
   (1 row)

   postgres=#
   ```

3. Exit from the database.

4. Check if the **pg_hba.conf** file has the entry **host all all 0.0.0.0/0 md5**, required to access all incoming connections. In addition, add the entry at the end of the **pg_hba.conf** file.

   Example: pg_hba.conf file

   ```
   sh-4.4$ cat /pgdata/pg14/pg_hba.conf

   # Do not edit this file manually!
   # It will be overwritten by Patroni!
   local all "postgres" peer
   hostssl replication "_crunchyrepl" all cert
   hostssl "postgres" "_crunchyrepl" all cert
   host all "_crunchyrepl" all reject
   hostssl all all all md5
   host  all  all 0.0.0.0/0 md5
   ```

5. Save the **pg_hba.conf** file and reload the database.

   ```
   sh-4.4$ psql -U postgres
   psql (14.4)
   Type "help" for help.

   postgres=# SHOW hba_file;
           hba_file
   --------------------------
    /pgdata/pg14/pg_hba.conf
   (1 row)
   ```

```
postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
```

6. Exit the database.

7. Decode the secret value of the Crunchy Postgres host.

   Example: Decode the secret value of a Crunchy Postgres host

   ```
   $ echo 'aGlwcG8tcHJpbWFyeS5vcGVuc2hpZnQtb3BlcmF0b3JzLnN2YyA=' | base64 --
   decode
   test-primary.openshift-operators.svc
   ```

8. Create a secret named **tekton-hub-db** in the target namespace with the following keys:

   - **POSTGRES_HOST**
   - **POSTGRES_DB**
   - **POSTGRES_USER**
   - **POSTGRES_PASSWORD**
   - **POSTGRES_PORT**

     Example: Custom database secrets

     ```
     apiVersion: v1
     kind: Secret
     metadata:
       name: tekton-hub-db
       labels:
         app: tekton-hub-db
     type: Opaque
     stringData:
       POSTGRES_HOST: test-primary.openshift-operators.svc
       POSTGRES_DB: test
       POSTGRES_USER: <username>
       POSTGRES_PASSWORD: <password>
       POSTGRES_PORT: '5432'
     ...
     ```

NOTE

The default target namespace is **openshift-pipelines**.

9. In the **TektonHub** CR, set the value of the database secret attribute to **tekton-hub-db**.

   Example: Adding custom database secret

   ```
   apiVersion: operator.tekton.dev/v1alpha1
   ```

```
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines
  db:
    secret: tekton-hub-db
...
```

10. Use the updated **TektonHub** CR to associate the custom database with Tekton Hub.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

11. Check the status of the installation. The **TektonHub** CR might take some time to attain a steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                  UIURL
hub    v1.9.0    True             https://api.route.url/  https://ui.route.url/
```

1.2.2. Optional: Migrating Tekton Hub data to an existing Crunchy Postgres database

Tekton Hub supports the use of Crunchy Postgres as a custom database. For a pre-installed Tekton Hub with default database, cluster administrators can use Crunchy Postgres as a custom database after migrating the Tekton Hub data from the internal or default database to the external Crunchy Postgres database.

Procedure

1. Dump the existing data from the internal or default database into a file in the pod.

Example: Dump data

```
$ pg_dump -Ft -h localhost -U postgres hub -f /tmp/hub.dump
```

2. Copy the file containing the data dump to your local system.

Command format

```
$ oc cp -n <namespace> <podName>:<path-to-hub.dump> <path-to-local-system>
```

Example

```
$ oc cp -n openshift-pipelines tekton-hub-db-7d6d888c67-p7mdr:/tmp/hub.dump
/home/test_user/Downloads/hub.dump
```

3. Copy the file that contains the data dump from the local system to the pod running the external Crunchy Postgres database.

Command format

```
$ oc cp -n <namespace> <path-to-local-system> <podName>:<path-to-hub.dump>
```

Example

```
$ oc cp -n openshift-operators /home/test_user/Downloads/hub.dump test-instance1-spnz-0:/tmp/hub.dump
```

4. Restore the data in the Crunchy Postgres database.

Command format

```
$ pg_restore -d <database-name> -h localhost -U postgres <path-where-file-is-copied>
```

Example

```
$ pg_restore -d test -h localhost -U postgres /tmp/hub.dump
```

5. Get into the Crunchy Postgres pod. .Example: Get into the **test-instance1-m7hh-0** pod

```
$ oc exec -it -n openshift-operators test-instance1-m7hh-0 -- /bin/sh

Defaulting container name to database.
Use 'oc describe pod/test-instance1-m7hh-0 -n openshift-operators' to see all of the containers in this pod.
sh-4.4$ psql -U postgres
psql (14.4)
Type "help" for help.
```

6. Find the **pg_hba.conf** file.

```
postgres=# SHOW hba_file;
         hba_file
--------------------------
 /pgdata/pg14/pg_hba.conf
(1 row)

postgres=#
```

7. Exit from the database.

8. Check if the **pg_hba.conf** file has the entry **host all all 0.0.0.0/0 md5**, which is necessary for accessing all incoming connections. If necessary, add the entry at the end of the **pg_hba.conf** file.

Example: pg_hba.conf file

```
sh-4.4$ cat /pgdata/pg14/pg_hba.conf

# Do not edit this file manually!
# It will be overwritten by Patroni!
local all "postgres" peer
```

```
hostssl replication "_crunchyrepl" all cert
hostssl "postgres" "_crunchyrepl" all cert
host all "_crunchyrepl" all reject
hostssl all all all md5
host  all  all 0.0.0.0/0 md5
```

9. Save the **pg_hba.conf** file and reload the database.

```
sh-4.4$ psql -U postgres
psql (14.4)
Type "help" for help.

postgres=# SHOW hba_file;
         hba_file
--------------------------
 /pgdata/pg14/pg_hba.conf
(1 row)

postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
```

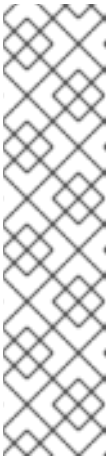
10. Exit the database.

11. Verify that a secret named **tekton-hub-db** in the target namespace has the following keys:

    - **POSTGRES_HOST**
    - **POSTGRES_DB**
    - **POSTGRES_USER**
    - **POSTGRES_PASSWORD**
    - **POSTGRES_PORT**

    Example: Custom database secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: tekton-hub-db
  labels:
    app: tekton-hub-db
type: Opaque
stringData:
  POSTGRES_HOST: test-primary.openshift-operators.svc
  POSTGRES_DB: test
  POSTGRES_USER: test
  POSTGRES_PASSWORD: woXOisU5>ocJiTF7y{{;1[Q(
  POSTGRES_PORT: '5432'
...
```



NOTE

The value of the **POSTGRES_HOST** field is encoded as a secret. You can decode the value of the Crunchy Postgres host by using the following example.

Example: Decode the secret value of a Crunchy Postgres host

```
$ echo
'aGlwcG8tcHJpbWFyeS5vcGVuc2hpZnQtb3BlcmF0b3JzLnN2YyA=' |
base64 --decode
test-primary.openshift-operators.svc
```

12. Verify that in the **TektonHub** CR, the value of the database secret attribute is **tekton-hub-db**.

Example: TektonHub CR with the name of the database secret

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines
  db:
    secret: tekton-hub-db
...
```

13. To associate the external Crunchy Postgres database with Tekton Hub, replace any existing **TektonHub** CR with the updated **TektonHub** CR.

```
$ oc replace -f <updated-tekton-hub-cr>.yaml
```

14. Check the status of the Tekton Hub. The updated **TektonHub** CR might take some time to attain a steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

1.3. UPDATING TEKTON HUB WITH CUSTOM CATEGORIES AND CATALOGS

Cluster administrators can update Tekton Hub with custom categories, catalogs, scopes, and default scopes that reflect the context of their organization.

Procedure

1. Optional: Edit the **categories**, **catalogs**, **scopes**, and **default:scopes** fields in the Tekton Hub CR.

NOTE

The default information for categories, catalog, scopes, and default scopes are pulled from the Tekton Hub API config map. If you provide custom values in the **TektonHub** CR, it overrides the default values.

2. Apply the Tekton Hub CR.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

3. Observe the Tekton Hub status.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url  https://ui.route.url
```

1.4. MODIFYING THE CATALOG REFRESH INTERVAL OF TEKTON HUB

The default catalog refresh interval for Tekton Hub is 30 minutes. Cluster administrators can modify the automatic catalog refresh interval by modifying the value of the **catalogRefreshInterval** field in the **TektonHub** CR.

Procedure

1. Modify the value of the **catalogRefreshInterval** field in the **TektonHub** CR.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonHub
metadata:
  name: hub
spec:
  targetNamespace: openshift-pipelines (1)
  api:
    catalogRefreshInterval: 30m (2)
```

(1) The namespace where Tekton Hub is installed; default is **openshift-pipelines**.

(2) The time interval after which the catalog refreshes automatically. The supported units of time are seconds (**s**), minutes (**m**), hours (**h**), days (**d**), and weeks (**w**). The default interval is 30 minutes.

2. Apply the **TektonHub** CR.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

3. Check the status of the installation. The **TektonHub** CR might take some time to attain steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

1.5. ADDING NEW USERS IN TEKTON HUB CONFIGURATION

Cluster administrators can add new users to Tekton Hub with different scopes.

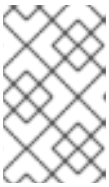
Procedure

1. Modify the **TektonHub** CR to add new users with different scopes.

```
...
scopes:
  - name: agent:create
    users: [<username_1>, <username_2>] (1)
  - name: catalog:refresh
    users: [<username_3>, <username_4>]
  - name: config:refresh
    users: [<username_5>, <username_6>]

default:
  scopes:
    - rating:read
    - rating:write
...
```

(1) The usernames registered with the Git repository hosting service provider.



NOTE

A new user signing in to Tekton Hub for the first time will have only the default scope. To activate additional scopes, ensure the user's username is added in the **scopes** field of the **TektonHub** CR.

2. Apply the updated **TektonHub** CR.

```
$ oc apply -f <tekton-hub-cr>.yaml
```

3. Check the status of the Tekton Hub. The updated **TektonHub** CR might take some time to attain a steady state.

```
$ oc get tektonhub.operator.tekton.dev
```

Sample output

```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/   https://ui.route.url/
```

4. Refresh the configuration.

```
$ curl -X POST -H "Authorization: <access-token>" \ 1
    --header "Content-Type: application/json" \
    --data '{"force": true} \
    <api-route>/system/config/refresh
```

1 The JWT token.

1.6. DISABLING TEKTON HUB AUTHORIZATION AFTER UPGRADING THE RED HAT OPENSHIFT PIPELINES OPERATOR FROM 1.7 TO 1.8

When you install Tekton Hub with Red Hat OpenShift Pipelines Operator 1.8, the login authorization and ratings for the Tekton Hub artifacts are disabled for the default installation. However, when you upgrade the Operator from 1.7 to 1.8, the instance of the Tekton Hub on your cluster does not automatically disable the login authorization and ratings.

To disable login authorization and ratings for Tekton Hub after upgrading the Operator from 1.7 to 1.8, perform the steps in the following procedure.

Prerequisites

- Ensure that the Red Hat OpenShift Pipelines Operator is installed in the default **openshift-pipelines** namespace on the cluster.

Procedure

1. Delete the existing Tekton Hub API secret that you created while manually installing Tekton Hub for Operator 1.7.

```
$ oc delete secret tekton-hub-api -n <targetNamespace> 1
```

1 The common namespace for the Tekton Hub API secret and the Tekton Hub CR. By default, the target namespace is **openshift-pipelines**.

2. Delete the **TektonInstallerSet** object for the Tekton Hub API.

```
$ oc get tektoninstallerset -o name | grep tekton-hub-api | xargs oc delete
```

NOTE

After deletion, the Operator automatically creates a new Tekton Hub API installer set.

Wait and check the status of the Tekton Hub. Proceed to the next steps when the **READY** column displays **True**.

```
$ oc get tektonhub hub
```

Sample output

```
NAME   VERSION      READY   REASON   APIURL
UIURL
hub    1.8.0        True             https://tekton-hub-api-openshift-pipelines.apps.example.com
https://tekton-hub-ui-openshift-pipelines.apps.example.com
```

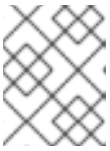
3. Delete the **ConfigMap** object for the Tekton Hub UI.

```
$ oc delete configmap tekton-hub-ui -n <targetNamespace> 1
```

1 The common namespace for the Tekton Hub UI and the Tekton Hub CR. By default, the target namespace is **openshift-pipelines**.

4. Delete the **TektonInstallerSet** object for the Tekton Hub UI.

```
$ oc get tektoninstallerset -o name | grep tekton-hub-ui | xargs oc delete
```



NOTE

After deletion, the Operator automatically creates a new Tekton Hub UI installer set.

Wait and check the status of the Tekton Hub. Proceed to the next steps when the **READY** column displays **True**.

```
$ oc get tektonhub hub
```

Sample output

```
NAME   VERSION      READY   REASON   APIURL
UIURL
hub    1.8.0        True             https://tekton-hub-api-openshift-pipelines.apps.example.com
https://tekton-hub-ui-openshift-pipelines.apps.example.com
```

1.7. ADDITIONAL RESOURCES

- GitHub repository of Tekton Hub
- Installing OpenShift Pipelines
- Red Hat OpenShift Pipelines release notes