Red Hat OpenShift Pipelines 1.13

自定义 Tekton Hub 实例

安装 Tekton Hub 的自定义实例

Last Updated: 2024-07-12

Red Hat OpenShift Pipelines 1.13 自定义 Tekton Hub 实例

安装 Tekton Hub 的自定义实例

法律通告

摘要

本文档提供有关安装和部署 Tekton Hub 自定义实例的信息。

目录

第 1 章 在 OPENSHIFT PIPELINES 中使用 TEKTON HUB

重要

Tekton Hub 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

Tekton Hub 可帮助您发现、搜索和共享 CI/CD 工作流可重复使用的任务和管道。Tekton Hub 的一个公共实例位于 hub.tekton.dev 中。集群管理员还可以通过修改 TektonHub 自定义资源(CR)中的配置来安装和部署 **Tekton Hub** 的自定义实例。

1.1. 在 OPENSHIFT CONTAINER PLATFORM 集群上安装并部署 TEKTON HUB

Tekton Hub 是一个可选组件；集群管理员无法使用 **TektonConfig** 自定义资源(CR)安装它。要安装和管理 Tekton Hub，请使用 **TektonHub** CR。

您可以使用以下模式在集群中安装 Tekton Hub：

- *没有* Tekton Hub 工件的登录授权和评级
- *使用* Tekton Hub 工件的登录授权和评级

注意

如果您使用 Github Enterprise 或 Gitlab Enterprise，请在与企业服务器相同的网络中安装并部署 Tekton Hub。例如，如果企业服务器在 VPN 后面运行，请在 VPN 之后也部署 Tekton Hub。

1.1.1. 在不登录和评级的情况下安装 Tekton Hub

您可以使用默认配置在集群中安装 Tekton Hub。在使用默认配置时，Tekton Hub 不支持使用 Tekton Hub 工件的授权和评级登录。

先决条件

- 确保在集群中的默认 **openshift-pipelines** 命名空间中安装了 Red Hat OpenShift Pipelines Operator。

流程

1. 创建一个类似以下示例的 **TektonHub** CR。

```
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) 需要安装 Tekton Hub 的命名空间；默认为 **openshift-pipelines**。

(2) 目录自动刷新的时间间隔。支持的时间单位为秒(**s**)、分钟(**m**)、小时(**h**)、天(**d**)和周(**w**)。默认间隔为 30 分钟。

注意

如果没有为 **TektonHub** CR 中的可选字段提供自定义值，则会使用 Tekton Hub API 配置映射中配置的默认值。

2. 应用 **TektonHub** CR。

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

3. 检查安装的状态。**TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

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

1.1.2. 使用登录和评级安装 Tekton Hub

您可以使用支持使用 Tekton Hub 工件的授权和评级登录的自定义配置在集群中安装 Tekton Hub。

先决条件

- 确保在集群中的默认 **openshift-pipelines** 命名空间中安装了 Red Hat OpenShift Pipelines Operator。

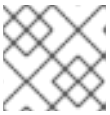
流程

1. 使用托管供应商的 Git 存储库创建 OAuth 应用程序，并记下客户端 ID 和客户端 Secret。支持的提供程序是 GitHub、GitLab 和 BitBucket。
   - 对于 GitHub OAuth 应用程序，请将 Homepage URL 和 Authorization 回调 URL 设置为 **<auth-route>**。
   - 对于 GitLab OAuth 应用程序，将 **REDIRECT_URI** 设置为 **<auth-route>/auth/gitlab/callback**。
   - 对于 BitBucket OAuth 应用，将 **Callback URL** 设置为 **<auth-route>**。
2. 编辑 **<tekton_hub_root>/config/02-api/20-api-secret.yaml** 文件，使其包含 Tekton Hub API secret。例如：

```
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. GitHub OAuth 应用程序的客户端 ID。
2. GitHub OAuth 应用程序的客户端 Secret。
3. GitLab OAuth 应用的客户端 ID。

4. GitLab OAuth 应用中的 Client Secret。
5. BitBucket OAuth 应用程序的客户端 ID。
6. BitBucket OAuth 应用程序的客户端机密。
7. 用于为用户创建的 JSON Web Token (JWT) 签名的随机字符串。
8. 添加访问令牌过期的时间限制。例如： **1m**，其中 m 表示分钟。支持的时间单位为秒(**s**)、分钟(**m**)、小时(**h**)、天(**d**)和周(**w**)。
9. 添加刷新令牌过期的时间限制。例如，**1m**，其中 **m** 表示分钟。支持的时间单位为秒(**s**)、分钟(**m**)、小时(**h**)、天(**d**)和周(**w**)。确保为令牌刷新设置的到期时间大于为令牌访问设置的到期时间。
10. OAuth 应用的路由 URL。
11. GitHub Enterprise URL，如果您使用 GitHub Enterprise 进行身份验证。不要提供目录的 URL 作为此字段的值。
12. GitLab Enterprise URL，如果您使用 GitLab Enterprise 进行身份验证。不要提供目录的 URL 作为此字段的值。



注意

您可以删除与部署无关的 Git 存储库托管服务供应商的未使用字段。

3. 创建一个类似以下示例的 **TektonHub** CR。

```
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. 需要安装 Tekton Hub 的命名空间；默认为 **openshift-pipelines**。
2. 可选：自定义数据库，如 Crunchy Postgres 数据库。
3. 数据库 secret 的名称必须是 **tekton-hub-db**。
4. 可选：为 Tekton Hub 中的任务和管道自定义类别。
5. 可选：为 Tekton Hub 自定义目录。
6. 可选：其他用户。您可以满足多个用户，如 **[<username_1>, <username_2>, <username_3>]**。
7. 可选：自定义默认范围。
8. 目录自动刷新的时间间隔。支持的时间单位为秒(**s**)、分钟(**m**)、小时(**h**)、天(**d**)和周(**w**)。默认间隔为 30 分钟。

注意

如果没有为 **TektonHub** CR 中的可选字段提供自定义值，则会使用 Tekton Hub API 配置映射中配置的默认值。

4. 应用 **TektonHub** CR。

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

5. 检查安装的状态。**TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

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

1.2. 可选：在 TEKTON HUB 中使用自定义数据库

集群管理员可以将自定义数据库用于 Tekton Hub，而不是 Operator 安装的默认 PostgreSQL 数据库。您可以在安装时关联自定义数据库，并将其与 Tekton Hub 提供的 **db-migration**、**api** 和 **ui** 接口一起使用。或者，即使安装了默认数据库，也可以将自定义数据库与 Tekton Hub 关联。

流程

1. 在目标命名空间中创建名为 **tekton-hub-db** 的 secret，其键如下：
   - **POSTGRES_HOST**
   - **POSTGRES_DB**
   - **POSTGRES_USER**
   - **POSTGRES_PASSWORD**
   - **POSTGRES_PORT**

   示例：自定义数据库 secret

```
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>
...
```

注意

默认目标命名空间是 **openshift-pipelines**。

2. 在 **TektonHub** CR 中，将 database secret 属性的值设置为 **tekton-hub-db**。

   示例：添加自定义数据库 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. 使用更新的 **TektonHub** CR 将自定义数据库与 Tekton Hub 关联。

   a. 如果您要在集群中安装 Tekton Hub 时关联自定义数据库，请应用更新的 **TektonHub** CR。

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

   b. 另外，如果您要在 Tekton Hub 安装完成后关联自定义数据库，请将现有的 **TektonHub** CR 替换为更新的 **TektonHub** CR。

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

4. 检查安装的状态。**TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

   输出示例

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

1.2.1. 可选：安装 Crunchy Postgres 数据库和 Tekton Hub

集群管理员可以安装 Crunchy Postgres 数据库，并将 Tekton Hub 配置为使用它，而不是默认数据库。

先决条件

- 从 Operator Hub 安装 Crunchy Postgres Operator。
- 创建一个 Postgres 实例，用于启动 Crunchy Postgres 数据库。

流程

1. 进入 Crunchy Postgres pod。

   示例：获取 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. 查找 **pg_hba.conf** 文件。

   ```
   postgres=# SHOW hba_file;
           hba_file
   ```

```
-------------------------
 /pgdata/pg14/pg_hba.conf
(1 row)

postgres=#
```

3. 退出数据库。

4. 检查 **pg_hba.conf** 文件是否有条目 **host all 0.0.0.0/0 md5**，以访问所有进入的连接。另外，在 **pg_hba.conf** 文件的末尾添加该条目。

   示例： pg_hba.conf 文件

```
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. 保存 **pg_hba.conf** 文件并重新载入数据库。

```
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. 退出数据库。

7. 解码 Crunchy Postgres 主机的 secret 值。

   示例：对 Crunchy Postgres 主机的 secret 值进行编码

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

8. 在目标命名空间中创建名为 **tekton-hub-db** 的 secret，其键如下：

   - **POSTGRES_HOST**

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

示例：自定义数据库 secret

```
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'
...
```

注意

默认目标命名空间是 **openshift-pipelines**。

9. 在 **TektonHub** CR 中，将 database secret 属性的值设置为 **tekton-hub-db**。

示例：添加自定义数据库 secret

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

10. 使用更新的 **TektonHub** CR 将自定义数据库与 Tekton Hub 关联。

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

11. 检查安装的状态。**TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

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

1.2.2. 可选：将 Tekton Hub 数据迁移到现有的 Crunchy Postgres 数据库

Tekton Hub 支持使用 Crunchy Postgres 作为自定义数据库。对于带有默认数据库的预安装的 Tekton Hub，集群管理员可在将 Tekton Hub 数据从内部或默认数据库迁移到外部 Crunchy Postgres 数据库后，使用 Crunchy Postgres 作为自定义数据库。

流程

1. 将内部或默认数据库的现有数据转储到 pod 中的文件中。

   示例：转储数据

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

2. 将包含数据转储的文件复制到您的本地系统中。

   命令格式

   ```
   $ 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. 将包含本地系统的数据转储的文件复制到运行外部 Crunchy Postgres 数据库的 pod。

   命令格式

   ```
   $ 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. 恢复 Crunchy Postgres 数据库中的数据。

   命令格式

   ```
   $ 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. 进入 Crunchy Postgres pod。例如: 获取 **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. 查找 **pg_hba.conf** 文件。

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

postgres=#
```

7. 退出数据库。

8. 检查 **pg_hba.conf** 文件是否有条目 **host all 0.0.0.0/0 md5**，这是访问所有传入连接所必需的。如有必要，在 **pg_hba.conf** 文件的末尾添加该条目。

   示例： pg_hba.conf 文件

```
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. 保存 **pg_hba.conf** 文件并重新载入数据库。

```
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. 退出数据库。

11. 验证目标命名空间中名为 **tekton-hub-db** 的 secret 具有以下键：

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

    示例：自定义数据库 secret

```
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'
...
```

注意

POSTGRES_HOST 字段的值编码为 secret。您可以使用以下示例解码 Crunchy Postgres 主机的值。

示例：对 Crunchy Postgres 主机的 secret 值进行编码

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

12. 验证在 **TektonHub** CR 中，数据库 secret 属性的值是否为 **tekton-hub-db**。

    示例： TektonHub CR，带有数据库 secret 的名称

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

```
  db:
    secret: tekton-hub-db
...
```

13. 要将外部 Crunchy Postgres 数据库与 Tekton Hub 关联，请将任何现有的 **TektonHub** CR 替换为更新的 **TektonHub** CR。

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

14. 检查 Tekton Hub 的状态。更新的 **TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

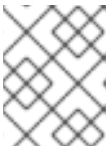
```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/    https://ui.route.url/
```

1.3. 使用自定义类别和目录更新 TEKTON HUB

集群管理员可以使用自定义类别、目录、范围和默认范围来更新 Tekton Hub。

流程

1. 可选：编辑 Tekton Hub CR 中的 **categories**, **catalogs**, **scopes**, 和 **default:scopes** 字段。



注意

类别、目录、范围和默认范围的默认信息是从 Tekton Hub API 配置映射中拉取的。如果您在 **TektonHub** CR 中提供自定义值，它会覆盖默认值。

2. 应用 Tekton Hub CR。

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

3. 观察 Tekton Hub 状态。

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

输出示例

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

1.4. 修改 TEKTON HUB 的目录刷新间隔

Tekton Hub 的默认目录刷新间隔为 30 分钟。集群管理员可以通过修改 **TektonHub** CR 中的 **catalogRefreshInterval** 字段的值来修改自动目录刷新间隔。

流程

1. 修改 **TektonHub** CR 中的 **catalogRefreshInterval** 字段的值。

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

(1) 安装 Tekton Hub 的命名空间；默认为 **openshift-pipelines**。

(2) 目录自动刷新的时间间隔。支持的时间单位为秒(**s**)、分钟(**m**)、小时(**h**)、天(**d**)和周(**w**)。默认间隔为 30 分钟。

2. 应用 **TektonHub** CR。

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

3. 检查安装的状态。**TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

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

1.5. 在 TEKTON HUB 配置中添加新用户

集群管理员可以使用不同范围向 Tekton Hub 添加新用户。

流程

1. 修改 **TektonHub** CR，以添加具有不同范围的新用户。

```
...
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 在 Git 存储库托管服务提供商中注册的用户名。

注意

第一次登录到 Tekton Hub 的新用户将只有默认范围。要激活其他范围，请确保在 **TektonHub** CR 的 **scopes** 字段中添加了用户的用户名。

2. 应用更新的 **TektonHub** CR。

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

3. 检查 Tekton Hub 的状态。更新的 **TektonHub** CR 可能需要一些时间才能获得 steady 状态。

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

输出示例

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

4. 刷新配置。

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

1 JWT 令牌.

1.6. 将 RED HAT OPENSHIFT PIPELINES OPERATOR 从 1.7 升级到 1.8 后禁用 TEKTON HUB 授权

当使用 Red Hat OpenShift Pipelines Operator 1.8 安装 Tekton Hub 时，默认安装禁用了 Tekton Hub 工件的登录授权和评级。但是，当您将 Operator 从 1.7 升级到 1.8 时，集群中的 Tekton Hub 实例不会自动禁用登录授权和评级。

要在将 Operator 从 1.7 升级到 1.8 后禁用 Tekton Hub 的登录授权和评级，请执行以下步骤。

先决条件

- 确保在集群中的默认 **openshift-pipelines** 命名空间中安装了 Red Hat OpenShift Pipelines Operator。

流程

1. 删除您在为 Operator 1.7 安装 Tekton Hub 时创建的现有 Tekton Hub API secret。

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

1 Tekton Hub API secret 和 Tekton Hub CR 的通用命名空间。默认情况下，目标命名空间是 **openshift-pipelines**

openshift-pipelines。

2. 删除 Tekton Hub API 的 **TektonInstallerSet** 对象。

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

注意

删除后，Operator 会自动创建一个新的 Tekton Hub API 安装程序设置。

等待并检查 Tekton Hub 的状态。当 **READY** 列显示为 **True** 时，继续执行后续步骤。

```
$ oc get tektonhub hub
```

输出示例

```
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. 删除 Tekton Hub UI 的 **ConfigMap** 对象。

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

1 Tekton Hub UI 和 Tekton Hub CR 的通用命名空间。默认情况下，目标命名空间是 **openshift-pipelines**。

4. 删除 Tekton Hub UI 的 **TektonInstallerSet** 对象。

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

注意

删除后，Operator 会自动创建一个新的 Tekton Hub UI 安装程序设置。

等待并检查 Tekton Hub 的状态。当 **READY** 列显示为 **True** 时，继续执行后续步骤。

```
$ oc get tektonhub hub
```

输出示例

```
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. 其他资源

- Tekton Hub的 GitHub 存储库
- 安装 OpenShift Pipelines
- Red Hat OpenShift Pipelines 发行注记