Red Hat OpenShift Pipelines 1.13

カスタム Tekton Hub インスタンス

Tekton Hub のカスタムインスタンスのインストール

Last Updated: 2023-12-12

Red Hat OpenShift Pipelines 1.13 カスタム Tekton Hub インスタンス

Tekton Hub のカスタムインスタンスのインストール

法律上の通知

概要

このドキュメントでは、Tekton Hub のカスタムインスタンスのインストールとデプロイに関する情報を提供します。

目次

第1章 OPENSHIFT PIPELINES での TEKTON HUB の使用

重要

Tekton Hub はテクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Tekton Hub は、CI/CD ワークフローの再利用可能なタスクとパイプラインを検出、検索、および共有するのに役立ちます。Tekton Hub のパブリックインスタンスは、hub.tekton.dev で利用できます。クラスター管理者は、**TektonHub** カスタムリソース (CR) の設定を変更することで、Tekton Hub のカスタムインスタンスをインストールしてデプロイすることもできます。

1.1. OPENSHIFT CONTAINER PLATFORM クラスターへの TEKTON HUB のインストールとデプロイ

Tekton Hub はオプションのコンポーネントです。クラスター管理者は、**TektonConfig** カスタムリソース (CR) を使用してこれをインストールできません。Tekton Hub をインストールおよび管理するには、**TektonHub** CR を使用します。

次の 2 つのモードを使用して、クラスターに Tekton Hub をインストールできます。

- Tekton Hub アーティファクトのログイン認証と評価 **なし**
- Tekton Hub アーティファクトのログイン認証と評価 **あり**

注記

Github Enterprise または Gitlab Enterprise を使用している場合は、エンタープライズサーバーと同じネットワークに Tekton Hub をインストールしてデプロイします。たとえば、エンタープライズサーバーが VPN の背後で実行されている場合は、同じく VPN の背後にあるクラスターに Tekton Hub をデプロイします。

1.1.1. ログインと評価なしで Tekton Hub をインストールする

Tekton Hub は、デフォルト設定でクラスターに自動的にインストールできます。デフォルト設定を使用する場合、Tekton Hub は Tekton Hub アーティファクトの認証と評価によるログインをサポートしません。

前提条件

- Red Hat OpenShift Pipelines Operator が、クラスターのデフォルトの **openshift-pipelines** namespace にインストールされている。

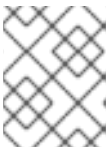
手順

1. 次の例のような **TektonHub** CR を作成します。

```
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) Tekton Hub をインストールする必要がある namespace。デフォルトは **openshift-pipelines** です。

(2) カタログが自動的に更新されるまでの時間間隔。サポートされている時間の単位は、秒 (**s**)、分 (**m**)、時間 (**h**)、日 (**d**)、および週 (**w**) です。デフォルトの間隔は 30 分です。



注記

TektonHub CR のオプションフィールドにカスタム値を指定しない場合は、Tekton Hub API config map で設定されたデフォルト値が使用されます。

2. **TektonHub** CR を適用します。

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

3. インストールのステータスを確認します。**TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

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

1.1.2. ログインと評価による Tekton Hub のインストール

Tekton Hub アーティファクトの承認と評価を使用したログインをサポートするカスタム設定を使用して、クラスターに Tekton Hub をインストールできます。

前提条件

- Red Hat OpenShift Pipelines Operator が、クラスターのデフォルトの **openshift-pipelines** namespace にインストールされている。

手順

1. Git リポジトリーホスティングプロバイダーを使用して OAuth アプリケーションを作成し、クライアント ID とクライアントシークレットをメモします。サポートされているプロバイダーは、GitHub、GitLab、および BitBucket です。
   - GitHub OAuth アプリケーション の場合、Homepage URL と Authorization callback URL を **<auth-route>** として設定します。
   - GitLab OAuth アプリケーション の場合は、**REDIRECT_URI** を **<auth-route>/auth/gitlab/callback** として設定します。
   - BitBucket OAuth アプリケーション の場合は、**Callback URL** を **<auth-route>** として設定します。
2. **<tekton_hub_root>/config/02-api/20-api-secret.yaml** ファイルを編集して、Tekton Hub API シークレットを含めます。以下に例を示します。

```
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 アプリケーションからのクライアント ID。
2. GitHub OAuth アプリケーションからのクライアントシークレット。
3. GitLab OAuth アプリケーションからのクライアント ID。
4. GitLab OAuth アプリケーションからのクライアントシークレット。
5. BitBucket OAuth アプリケーションからのクライアント ID。
6. BitBucket OAuth アプリケーションからのクライアントシークレット。
7. ユーザー用に作成された JSON Web トークン (JWT) に署名するために使用する長いランダムな文字列。
8. アクセストークンの有効期限が切れるまでの時間制限を追加します。たとえば **1m** とした場合、 m は分を示します。サポートされている時間の単位は、秒 (**s**)、分 (**m**)、時間 (**h**)、日 (**d**)、および週 (**w**) です。
9. 更新トークンの有効期限が切れるまでの時間制限を追加します。たとえば、**1m** とした場合、**m** は分を示します。サポートされている時間の単位は、秒 (**s**)、分 (**m**)、時間 (**h**)、日 (**d**)、および週 (**w**) です。トークンの更新に設定された有効期限が、トークンアクセスに設定された有効期限よりも長いことを確認してください。
10. OAuth アプリケーションのルート URL。
11. GitHub Enterprise を使用して認証している場合は、GitHub Enterprise URL。このフィールドの値としてカタログへの URL を指定しないでください。
12. GitLab Enterprise を使用して認証している場合は、GitLab Enterprise URL。このフィールドの値としてカタログへの URL を指定しないでください。

注記

デプロイに関係のない Git リポジトリーホスティングサービスプロバイダーの未使用のフィールドを削除できます。

3. 次の例のような **TektonHub** CR を作成します。

```
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. Tekton Hub をインストールする必要がある namespace。デフォルトは **openshift-pipelines** です。
2. オプション: Crunchy Postgres データベースなどのカスタムデータベース。
3. データベースシークレットの名前は **tekton-hub-db** である必要があります。
4. オプション: Tekton Hub のタスクとパイプラインのカスタマイズされたカテゴリー。
5. オプション: Tekton Hub 用にカスタマイズされたカタログ。
6. オプション: 追加のユーザー。**[<username_1>, <username_2>, <username_3>]** のように、複数のユーザーをメンションできます。
7. オプション: カスタマイズされたデフォルトスコープ。
8. カタログが自動的に更新されるまでの時間間隔。サポートされている時間の単位は、秒 (**s**)、分 (**m**)、時間 (**h**)、日 (**d**)、および週 (**w**) です。デフォルトの間隔は 30 分です。

注記

TektonHub CR のオプションフィールドにカスタム値を指定しない場合は、Tekton Hub API config map で設定されたデフォルト値が使用されます。

4. **TektonHub** CR を適用します。

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

5. インストールのステータスを確認します。**TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

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

1.2. オプション: TEKTON HUB でのカスタムデータベースの使用

クラスター管理者は、Operator によってインストールされたデフォルトの PostgreSQL データベースの代わりに、カスタムデータベースを Tekton Hub で使用できます。インストール時にカスタムデータベースを関連付けて、Tekton Hub が提供する **db-migration**、**api**、および **ui** インターフェイスで使用できます。または、デフォルトデータベースでのインストールが完了した後でも、カスタムデータベースを Tekton Hub に関連付けることができます。

手順

1. 次のキーを使用して、ターゲット namespace に **tekton-hub-db** という名前のシークレットを作成します。

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

   例: カスタムデータベースシークレット

```
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>
...
```

注記

デフォルトのターゲット namespace は **openshift-pipelines** です。

2. **TektonHub** CR で、データベースのシークレット属性の値を **tekton-hub-db** に設定します。

例: カスタムデータベースシークレットの追加

```
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. 更新された **TektonHub** CR を使用して、カスタムデータベースを Tekton Hub に関連付けます。

   a. クラスターに Tekton Hub をインストールするときにカスタムデータベースを関連付ける場合は、更新された **TektonHub** CR を適用します。

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

   b. または、Tekton Hub のインストールが完了した後にカスタムデータベースを関連付ける場合は、既存の **TektonHub** CR を更新された **TektonHub** CR に置き換えます。

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

4. インストールのステータスを確認します。**TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

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

1.2.1. オプション: Crunchy Postgres データベースと Tekton Hub のインストール

クラスター管理者は、Crunchy Postgres データベースをインストールし、デフォルトデータベースの代わりにそれを使用するように、Tekton Hub を設定できます。

前提条件

- Operator Hub から Crunchy Postgres Operator をインストールする。
- Crunchy Postgres データベースを起動する Postgres インスタンスを作成する。

手順

1. Crunchy Postgres Pod に入ります。

   例: test-instance1-m7hh-0 Pod に入ります

   ```
   $ 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. **pg_hba.conf** ファイルを見つけます。

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

   postgres=#
   ```

3. データベースを終了します。

4. **pg_hba.conf** ファイルに、すべての受信接続にアクセスするために必要なエントリー **host all all 0.0.0.0/0 md5** があるかどうかを確認します。さらに、**pg_hba.conf** ファイルの末尾にエントリーを追加します。

   例: pg_hba.conf ファイル

   ```
   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. **pg_hba.conf** ファイルを保存し、データベースをリロードします。

```
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. データベースを終了します。
7. Crunchy Postgres ホストのシークレット値をデコードします。

   例: Crunchy Postgres ホストのシークレット値をデコードします

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

8. 次のキーを使用して、ターゲット namespace に **tekton-hub-db** という名前のシークレットを作成します。

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

     例: カスタムデータベースシークレット

     ```
     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'
     ...
     ```

注記

デフォルトのターゲット namespace は **openshift-pipelines** です。

9. **TektonHub** CR で、データベースのシークレット属性の値を **tekton-hub-db** に設定します。

例: カスタムデータベースシークレットの追加

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

10. 更新された **TektonHub** CR を使用して、カスタムデータベースを Tekton Hub に関連付けます。

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

11. インストールのステータスを確認します。**TektonHub** CR が安定状態になるまでは、時間がかかる場合があります。

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

出力例

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

1.2.2. オプション: Tekton Hub データを既存の Crunchy Postgres データベースに移行する

Tekton Hub は、カスタムデータベースとして Crunchy Postgres の使用をサポートしています。デフォルトデータベースを備えたプリインストールされた Tekton Hub の場合、クラスター管理者は、Tekton Hub データを内部またはデフォルトのデータベースから外部の Crunchy Postgres データベースに移行した後、Crunchy Postgres をカスタムデータベースとして使用できます。

手順

1. 内部またはデフォルトのデータベースから Pod 内のファイルに既存のデータをダンプします。

例: データのダンプ

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

2. データダンプを含むファイルをローカルシステムにコピーします。

コマンドの形式

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

例

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

3. データダンプを含むファイルをローカルシステムから外部の Crunchy Postgres データベースを実行している Pod にコピーします。

コマンドの形式

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

例

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

4. Crunchy Postgres データベース内のデータを復元します。

コマンドの形式

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

例

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

5. Crunchy Postgres Pod に入ります。例: **test-instance1-m7hh-0** Pod に入ります

```
$ 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. **pg_hba.conf** ファイルを見つけます。

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

postgres=#
```

7. データベースを終了します。

8. **pg_hba.conf** ファイルに、すべての受信接続にアクセスするために必要なエントリー **host all all 0.0.0.0/0 md5** があるか確認します。必要に応じて、**pg_hba.conf** ファイルの末尾にエントリーを追加します。

   例: pg_hba.conf ファイル

```
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. **pg_hba.conf** ファイルを保存し、データベースをリロードします。

```
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. データベースを終了します。

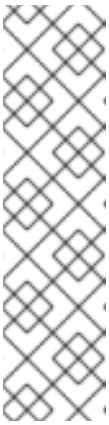
11. ターゲット namespace の **tekton-hub-db** という名前のシークレットに次のキーがあることを確認します。

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

      例: カスタムデータベースシークレット

```
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'
...
```



注記

POSTGRES_HOST フィールドの値はシークレットとしてエンコードされます。次の例を使用して、Crunchy Postgres ホストの値をデコードできます。

例: Crunchy Postgres ホストのシークレット値をデコードします

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

12. **TektonHub** CR で、データベースのシークレット属性の値が **tekton-hub-db** であることを確認します。

例: データベースシークレットの名前を含む TektonHub CR

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

13. 外部の Crunchy Postgres データベースを Tekton Hub に関連付けるには、既存の **TektonHub** CR を更新された **TektonHub** CR に置き換えます。

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

14. Tekton Hub のステータスを確認します。更新された **TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

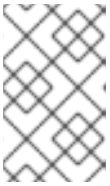
```
NAME   VERSION   READY   REASON   APIURL                 UIURL
hub    v1.9.0    True             https://api.route.url/    https://ui.route.url/
```

1.3. カスタムカテゴリーとカタログによる TEKTON HUB の更新

クラスター管理者は、自社のコンテキストを反映するカスタムカテゴリー、カタログ、スコープ、およびデフォルトスコープで Tekton Hub を更新できます。

手順

1. オプション: Tekton Hub CR の **category**、**catalogs**、**scopes**、および **default:scopes** フィールドを編集します。



注記

カテゴリー、カタログ、スコープ、およびデフォルトスコープのデフォルト情報は、Tekton Hub API config map から取得されます。**TektonHub** CR でカスタム値を指定すると、デフォルト値がオーバーライドされます。

2. Tekton Hub CR を適用します。

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

3. Tekton Hub のステータスを監視します。

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

出力例

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

1.4. TEKTON HUB のカタログ更新間隔の変更

Tekton Hub のデフォルトのカタログ更新間隔は 30 分です。クラスター管理者は、**TektonHub** CR の **catalogRefreshInterval** フィールドの値を変更することで、カタログの自動更新間隔を変更できます。

手順

1. **TektonHub** CR の **catalogRefreshInterval** フィールドの値を変更します。

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

(1) Tekton Hub がインストールされている namespace。デフォルトは **openshift-pipelines** です。

2 カタログが自動的に更新されるまでの時間間隔。サポートされている時間の単位は、秒 (**s**)、分 (**m**)、時間 (**h**)、日 (**d**)、および週 (**w**) です。デフォルトの間隔は 30 分です。

2. **TektonHub** CR を適用します。

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

3. インストールのステータスを確認します。**TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

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

1.5. TEKTON HUB 設定での新しいユーザーの追加

クラスター管理者は、さまざまなスコープで新しいユーザーを Tekton Hub に追加できます。

手順

1. **TektonHub** CR を変更して、異なるスコープを持つ新しいユーザーを追加します。

```
...
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 Git リポジトリーホスティングサービスプロバイダーに登録されているユーザー名。

注記

初めて Tekton Hub にサインインする新しいユーザーには、デフォルトのスコープのみが割り当てられます。追加のスコープを有効にするには、ユーザーのユーザー名が **TektonHub** CR の **scopes** フィールドに追加されていることを確認します。

2. 更新された **TektonHub** CR を適用します。

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

3. Tekton Hub のステータスを確認します。更新された **TektonHub** CR が安定状態になるまでに時間がかかる場合があります。

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

出力例

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

4. 設定を更新します。

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

1 JWT トークン。

1.6. RED HAT OPENSHIFT PIPELINES OPERATOR を 1.7 から 1.8 にアップグレードした後の TEKTON HUB 許可の無効化

Red Hat OpenShift Pipelines Operator 1.8 を使用して Tekton Hub をインストールすると、デフォルトのインストールでは、Tekton Hub アーティファクトのログイン認証と評価が無効になります。ただし、Operator を 1.7 から 1.8 にアップグレードすると、クラスター上の Tekton Hub のインスタンスは、ログイン認証と評価を自動的に無効にしません。

Operator を 1.7 から 1.8 にアップグレードした後、Tekton Hub のログイン認証と評価を無効にするには、次の手順を実行します。

前提条件

- Red Hat OpenShift Pipelines Operator が、クラスターのデフォルトの **openshift-pipelines** namespace にインストールされている。

手順

1. Operator 1.7 用の Tekton Hub を手動でインストールするときに作成した既存の Tekton Hub API シークレットを削除します。

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

1 Tekton Hub API シークレットと Tekton Hub CR の共通 namespace。デフォルトでは、ターゲット namespace は **openshift-pipelines** です。

2. Tekton Hub API の **TektonInstallerSet** オブジェクトを削除します。

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

注記

削除後、Operator は新しい Tekton Hub API インストーラーセットを自動的に作成します。

しばらく待って、Tekton Hub のステータスを確認してください。**READY** 列に **True** が表示されたら、次の手順に進みます。

```
$ oc get tektonhub hub
```

出力例

```
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. Tekton Hub UI の **ConfigMap** オブジェクトを削除します。

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

1 Tekton Hub UI と Tekton Hub CR の共通ネームスペース。デフォルトでは、ターゲット namespace は **openshift-pipelines** です。

4. Tekton Hub UI の **TektonInstallerSet** オブジェクトを削除します。

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

注記

削除後、Operator は新しい Tekton Hub UI インストーラーセットを自動的に作成します。

しばらく待って、Tekton Hub のステータスを確認してください。**READY** 列に **True** が表示されたら、次の手順に進みます。

```
$ oc get tektonhub hub
```

出力例

```
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. 関連情報

- Tekton Hub の GitHub リポジトリー。
- OpenShift Pipelines のインストール

- Red Hat OpenShift Pipelines リリースノート