Red Hat Trusted Application Pipeline 1.0

Installing Red Hat Trusted Application Pipeline

Learn how to install Red Hat Trusted Application Pipeline in your cluster.

Last Updated: 2024-06-17

Red Hat Trusted Application Pipeline 1.0 Installing Red Hat Trusted Application Pipeline

Learn how to install Red Hat Trusted Application Pipeline in your cluster.

Legal Notice

Abstract

This document provides information about how to install Red Hat Trusted Application Pipeline in your cluster.

Table of Contents

PREFACE ... 3
CHAPTER 1. CREATING A GITHUB APPLICATION FOR RHTAP ... 5
CHAPTER 2. FORKING THE RHTAP CATALOG REPOSITORY ... 8
CHAPTER 3. CREATING A GITOPS GIT TOKEN ... 9
CHAPTER 4. CREATING THE DOCKER CONFIGURATION VALUE ... 10
CHAPTER 5. CREATING A PRIVATE-VALUES.YAML FILE ... 11
CHAPTER 6. INSTALLING RHTAP IN YOUR CLUSTER ... 14
CHAPTER 7. FINALIZING YOUR GITHUB APPLICATION ... 15

PREFACE

Red Hat Trusted Application Pipeline (RHTAP) is not really a single product. Instead, it is a set of products that combine to form a highly automated, customizable, and secure platform for building applications.

RHTAP includes the following products:

- Red Hat Developer Hub : a self-service portal for developers.
- OpenShift GitOps: to manage Kubernetes deployments and their infrastructure.
- OpenShift Pipelines: to enable automation and provide visibility for continuous integration and continuous delivery (CI/CD) of software.
- Trusted Artifact Signer: to sign and validate the artifacts that RHTAP produces.
- Trusted Profile Analyzer: to deliver actionable information about your security posture.

It also depends on the following products:

- Quay.io: a container registry, where RHTAP stores your artifacts.
- Advanced Cluster Security (ACS) : a security tool that RHTAP uses to scan your artifacts.

NOTE

To see exactly which versions of these products RHTAP supports, reference the compatibility and support matrix in our Release notes.

Because a fully-operational instance of RHTAP involves all of the products listed above, installing RHTAP takes time and effort. However, we have automated this process where possible, and are providing instructions here that we hope are helpful and concise.

Additionally, be aware that the RHTAP installer is not a manager: it does not support upgrades. The installer generates your first deployment of RHTAP. After installation, you manage each product within RHTAP individually.

Before you can begin installation, you must meet six prerequisites. Then you must complete seven procedures.

Prerequisites

- ClusterAdmin access to an OpenShift Container Platform (OCP) cluster, through both the CLI and the web console
- An instance of Red Hat Advanced Cluster Security, as well as the following values from that instance:
    - ACS API token. You can follow the instructions for the prerequisites here to create an API token.
    - ACS central endpoint URL. You can follow the instructions here to configure the endpoint.
- To enable ACS to access private repositories in image registries, ACS will need to be configured for your specific registry

    - For Quay.io, under Integrations→Image Integrations select the Quay.io card
    - Add your OAUTH tokens to access your specific Quay.io instance
    - Validate the access via the test button. This will ensure if the RHTAP is asked to scan a private image, ACS will have access
- A Quay.io account
- The Helm CLI tool
- A GitHub account

Procedures

1. Creating a GitHub application for RHTAP
2. Forking the template catalog
3. Creating a GitOps git token
4. Creating the Docker configuration value
5. Creating a private-values.yaml file
6. Installing RHTAP in your cluster
7. Finalizing your GitHub application

The following pages of this document explain each of those procedures in detail. If you have the prerequisites, you are ready to start the installation process by creating a GitHub application.

CHAPTER 1. CREATING A GITHUB APPLICATION FOR RHTAP

Creating a GitHub application for RHTAP allows developers to authenticate to Red Hat Developer Hub, which is the user interface (UI) where they can use RHTAP. This GitHub application also allows RHTAP to access developer's source code that is hosted on GitHub.

Keep in mind that you must create and install the new application in a GitHub organization that you own and want to use for your instance of Red Hat Trusted Application Pipeline. RHTAP can subsequently create new repositories within that organization, to serve as the source code for the applications it builds.

Prerequisites

- Ownership of a GitHub organization

Procedure

1. Login to GitHub and go to your organizations (**Settings > Organizations**).
2. Click on an organization that you own and want to use for this instance of RHTAP. Or you can select **New organization** to create a new organization.
3. In the organization context, navigate to the GitHub Apps page (**Settings > Developer settings > GitHub Apps**).
4. Near the top banner, on the right side of the page, select **New GitHub App**.
5. If prompted, authenticate as needed.
6. In the **GitHub App name** field, enter a unique name.
7. In the **Homepage URL** field, enter a placeholder value, for example, https://www.placeholder.com.
8. In the **Callback URL** field, enter a placeholder value. You can use the same placeholder value, for example, https://www.placeholder.com.
9. In the **Webhook URL** field, enter a placeholder value. You can use the same placeholder value, for example, https://www.placeholder.com. Also, ensure that the **Active** checkbox is checked (GitHub should do this by default).
10. Create a new file on your local system, in which you save several values that you need for later steps in the installation process. When you enter values in this file, make sure to label them, so you can remember what each value is later on.

    ```
    $ touch ~/install_values.txt
    ```

11. In your CLI, generate a secret, then label and save it in **~/install_values.txt**.

    a. If you do not have OpenSSL, you can follow the download instructions.

       ```
       $ openssl rand -hex 20 >> ~/install_values.txt
       ```

IMPORTANT

Be sure to save the output of this command!

12. In GitHub, in the **Webook secret** field, enter the output of the last command.
13. Under **Repository permissions**, set the following permissions:
    a. **Administration: Read and write**
    b. **Checks: Read and write**
    c. **Contents: Read and write**
    d. **Issues: Read and write**
    e. **Metadata: Read-only** (this should already be set correctly, but verify its value)
    f. **Pull requests: Read and write**
14. Under **Organization permissions**, set the following permissions:
    a. **Members: Read-only**
    b. **Plan: Read-only**
15. Under **Subscribe to events**, select the following subscriptions:
    a. **Check run**
    b. **Check suite**
    c. **Commit comment**
    d. **Issue comment**
    e. **Pull request**
    f. **Push**
16. Under **Where can this GitHub App be installed?** select **Any account**.
17. Click **Create GitHub App**. You should then see the Developer Settings page.
18. Retrieve the Client ID and Application ID. Label and save them in your **~install_values.txt**.

IMPORTANT

The next two steps explain how to gather a client secret and a private key. You must save the client secret and private key, and keep them accessible, to complete the installation process for RHTAP!

19. On your new application's page, next to **Client secrets**, select **Generate a new client secret** Label and save the client secret, in **~/install_values.txt**.
20. On the same page in GitHub, under **Private keys**, select the **Generate a private key** button. Your system downloads a **private-key** file, which contains the private key. Label and save the

content of the private key file in **~/install_values.txt**. The private key should start with **-----BEGIN RSA PRIVATE KEY-----**, and end with **-----END RSA PRIVATE KEY-----**.

21. Still on the same page in GitHub, from the tabs on the left-hand side, select **Install App**.

22. Use the green **Install** button next to the name of your organization.

23. When prompted, select **All repositories**, so RHTAP can create new repositories in your organization. Click the green **Install** button.

Additional resources

- The procedure in this document is based on the Pipelines as Code documentation for creating a GitHub application.

CHAPTER 2. FORKING THE RHTAP CATALOG REPOSITORY

Once developers start using your instance of RHTAP, you might want to customize your instance, to better suit their needs. One aspect of RHTAP that you can customize is the set of software templates that it provides. These templates help developers quickly build applications.

Forking our catalog repository, which contains the default set of software templates, enables you to customize the templates for your instance.

Prerequisites:

- A GitHub account

Procedure:

1. In your web browser, navigate to the Releases page of the RHTAP software catalog repository.
2. Select the release that corresponds to the version of RHTAP that you are using.
   a. For example, if you are using version 1.0.0 of RHTAP, you should use this release.
3. Beneath the banner of the page for that release, select **Fork** to fork the repository.

NOTE

Be sure to update your fork from time to time, so updates from the upstream repository can benefit your instance of RHTAP.

CHAPTER 3. CREATING A GITOPS GIT TOKEN

In this procedure and the next one, you create two more values that you need in your **~/install_values.txt** file to complete installation.

Prerequisites:

- A GitHub account

Procedure:

1. In your web browser, go to your Developer Settings page in GitHub.
2. In the left panel, under **Personal access tokens**, select **Tokens (classic)**.
3. From the **Generate new token** drop down menu under the page banner, select **Generate new token (classic)**. You may need to authenticate to continue.
4. Enter a name, select an expiration date, and under **Select scopes**, select **repo** (which should automatically include all scopes from **repo: status** to **security_events**).
5. Select **Generate token**. GitHub redirects you to a new page, where your token is visible. Make sure to label and save this token in **~/install_values.txt**.

CHAPTER 4. CREATING THE DOCKER CONFIGURATION VALUE

In this procedure, you create the final value you need in your ~/**install_values.txt** file.

Prerequisites:

- A Quay.io account

Procedure:

1. In your web browser, login to Quay.io. In the right side of the banner, select your username and select **Account Settings** from the dropdown menu.

2. On your user settings page, under **Docker CLI Password**, select **Generate Encrypted Password**. In the popup window, enter your password to authenticate.

3. Next, still in the popup window, select **Docker Configuration** > **View [username]-auth.json** Copy the string, without the quotation marks, following **"auth":**.

4. In your ~/**install_values.txt** file, label and create the Docker configuration value with the following format, using your username and auth token where appropriate: {"auths": {"quay.io/[username]": {"auth": "[auth token]","email": ""}}}

NOTE

If you plan to use a private repository on Quay to host the images that RHTAP builds, you must also register that private repository with Advanced Cluster Security, as described in this document.

CHAPTER 5. CREATING A PRIVATE-VALUES.YAML FILE

RHTAP relies on Helm to automate much of the installation process. However, Helm requires specific information to install RHTAP correctly and in your cluster. You must provide that information in a file that you can reference in the install command. The file is called **private-values.yaml**.

This file is complex, and it could be easy to prepare it incorrectly. However, this procedure explains how to make the process of preparing **private-values.yaml** much simpler. It guides you to clone the RHTAP installer repository from GitHub and use a shell script within that repository, to generate **private-values.yaml** much more easily.

Prerequisites:

- git CLI tool
- yq CLI tool
- An **~/install_values.txt** file with all the necessary values. You created this file during the first, third, and fourth procedures.
- An API token and central endpoint of your instance of Advanced Cluster Security.

Procedure:

1. In your web browser, navigate to the RHTAP installer repository on GitHub.
2. Select the green **<> CODE** button. Under the Local tab, select your preferred connection type (HTTPS, SSH, or GitHub CLI) for cloning, and copy the given command.
3. Run the command you copied. For example, for SSH, run the following command in your CLI:

   ```
   $ git clone git@github.com:redhat-appstudio/rhtap-installer.git
   ```

4. In your CLI, navigate to your local clone of the RHTAP installer repository.

   ```
   $ cd rhtap-installer
   ```

5. Run the **bin/make.sh** script.

   ```
   $ bin/make.sh values
   ```

6. The script prompts you to enter values for each of the following fields. Follow the instructions to determine the value that you should enter. If you need to stop the script at any time, you may do so and simply rerun the **bin/make.sh values** command to resume your progress:

   a. **RHTAP_ENABLE_GITHUB**: Enter **y** if you want to use GitHub as a git repository for your applications

   b. **RHTAP_ENABLE_GITLAB**: Enter **y** if you want to use GitLab as a git repository for your applications

IMPORTANT

At the time of publication, our documentation does not explain how to configure GitLab as a git host for RHTAP, but it is possible. We are working on documenting that process. In the meantime, if you want to use GitLab, please reference the documentaiton provided at the end of this procedure.

c. **RHTAP_ENABLE_DEVELOPER_HUB**: Enter **y**

d. **RHTAP_ENABLE_TAS**: Enter **y** if you want to use Red Hat Trusted Artifact Signer to strengthen the security of your software supply chain.

e. **RHTAP_ENABLE_TAS_FULCIO_OIDC_DEFAULT_VALUES**: Enter **y** if you set the previous value to **y**.

f. **RHTAP_ENABLE_TPA**: Enter **y** if you want to use Red Hat Trusted Profile Analyzer to strengthen the security of your software supply chain.

g. **ACS__API_TOKEN**: Enter an API token for your ACS instance. You can follow the instructions for the prerequisites here to create an API token.

h. **ACS__CENTRAL_ENDPOINT**: Enter the endpoint of your ACS instance. You can follow the instructions here to configure the endpoint.

i. **DEVELOPER_HUB__CATALOG__URL**: Enter the address of the **all.yaml** file in your fork that you created in the last procedure.

```
https://github.com/[username]/tssc-sample-templates/blob/main/all.yaml
```

j. **GITHUB__APP__ID**: This value should be in your **~/install_values.txt** file. You saved it during the first installation procedure.

k. **GITHUB__APP__CLIENT__ID**: This value should be in your **~/install_values.txt** file. You saved it during the the first installation procedure.

l. **GITHUB__APP__CLIENT__SECRET**: This value should be in your **~/install_values.txt** file. You created and saved it during the first procedure.

m. **GITHUB__APP__PRIVATE_KEY**: This value should be in your **~/install_values.txt** file. You created and saved it during the first procedure.

n. **GITHUB__APP__WEBHOOK__SECRET**: This value should be in your **~/install_values.txt** file. You created and saved it during the first procedure.

o. **GITOPS__GIT_TOKEN**: This value should be in your **~/install_values.txt** file. You created and saved it in the second procedure.

p. **QUAY__API_TOKEN**: Use 'null' if your image repository is public. Otherwise create an API token with read access and paste its value.

q. **QUAY__DOCKERCONFIGJSON**: This value should be in your **~/install_values.txt** file. You created and saved it in the last procedure.

r. **TAS__SECURESIGN__FULCIO__ORG_EMAIL**: Enter the email of the person or team at your organization who owns this new instance of RHTAP.

s. **TAS__SECURESIGN__FULCIO__ORG_NAME**: Enter your GitHub organization's name.

> **NOTE**
>
> The remaining values are passwords and secrets that you must generate. You do not necessarily have to save these values elsewhere, since the script you are currently using is creating a file that stores all the values you enter.

t. **TPA__GUAC__PASSWORD**: Enter a strong password that you and your team members can use to validate yourselves for TPA's GUAC. You can use the same OpenSSL command you used previously to create a Webhook secret.

```
$ openssl rand -hex 20
```

u. **TPA__KEYCLOAK__ADMIN_PASSWORD**: Enter another strong password.

v. **TPA__MINIO__ROOT_PASSWORD**: Enter another strong password.

w. **TPA__OIDC__TESTING_MANAGER_CLIENT_SECRET**: Enter another value that can be used as a secure secret. You can use the OpenSSL command to generate this value as well.

x. **TPA__OIDC__TESTING_USER_CLIENT_SECRET**: Enter another value that can be used as a secure secret.

y. **TPA__OIDC__WALKER_CLIENT_SECRET**: Enter another value that can be used as a secure secret.

z. **TPA__POSTGRES__POSTGRES_PASSWORD**: Enter another strong password.

aa. **TPA__POSTGRES__TPA_PASSWORD**: Enter another strong password.

7. (Optional) After running **bin/make.sh**, you can change which namespace RHTAP uses to deploy applications. In the same context where you ran **bin/make.sh**, look for your newly generated **private-values.yaml** file. Open that file, and edit or add other namespaces under **namespaces:**, where currently you should see **- rhtap-app**.

Additional resources

If you wish to use GitLab as a git host before our documentation on that install path is complete, please reference the following documents:

- Configure GitLab as an OAuth 2.0 authentication identity provider
- Using Pipelines as Code with GitLab

CHAPTER 6. INSTALLING RHTAP IN YOUR CLUSTER

Once you have created a GitHub application and a private values.yaml file, you are ready to install RHTAP. The actual installation process is quite simple.

Prerequisites

- ClusterAdmin access to OCP cluster via CLI
- A **private-values.yaml** file (generated during the previous procedure)

Procedure

1. In your CLI, access your OCP cluster as ClusterAdmin.

   ```
   $  oc login [cluster address] -u kubeadmin -p [password]
   ```

2. Add the installer Helm repository to your local system.

   ```
   $ helm repo add openshift-helm-charts https://charts.openshift.io/
   ```

   a. If you already added this Helm repository earlier, update your Helm repositories.

      ```
      $ helm repo update
      ```

3. In the directory where you ran **bin/make.sh**, run the install command. Installation may take ten minutes or longer to complete.

   ```
   $ helm upgrade installer openshift-helm-charts/redhat-trusted-application-pipeline --install --
   create-namespace --namespace rhtap --timeout 20m --values private-values.yaml
   ```

   NOTE

   After **--namespace** you can specify any namespace you want; you do not have to use **rhtap**.

4. Once installation is complete, the installer provides output. Label and save both PipelineRun files that the installer generates in **~/install_values.txt**.

5. Copy and paste the entire first PipelineRun, from **cat << EOF | kubectl create** to **EOF** into your command line. Press enter to execute this PipelineRun.

6. Once the PipelineRun completes, it provides output which includes a link. If you are not already logged into the OpenShift web console, login first. Then open the link from the PipelineRun output.

Troubleshooting

- If Helm is unable to successfully install RHTAP for any reason, try deleting the **rhatp** project in your cluster, and running the install command again.

CHAPTER 7. FINALIZING YOUR GITHUB APPLICATION

After installing RHTAP, you must replace the placeholder values you previously entered in your GitHub application with values specific to your cluster. This allows anyone who installs the GitHub application to authenticate to Red Hat Developer Hub, and use Red Hat Trusted Application Pipeline.

Prerequisites:

- ClusterAdmin access to an OpenShift cluster via the web console

Procedure:

1. By opening the link generated at the end of the last procedure, you should be in the OpenShift console, using the **Administrator** view. If not, navigate to view in the OpenShift Console for your cluster.
    a. Use the left navigation bar to go to **Pipelines > Pipelines**.
    b. In the **Project** field, below the banner of the page, select **rhtap**.
    c. Select the **PipelineRun** tab. Select the PipelineRun with a name that starts with **rhtap-pe-info-**.
    d. Navigate to the **Logs** tab.
2. In a separate browser tab, return to the GitHub Apps page (**Settings > Developer settings > GitHub Apps**).
3. Next to your new custom application, click **Edit**.
4. Replace the placeholder values for the following fields with the new values found in the logs of the PipelineRun you executed in the OpenShift console:
    a. **Homepage URL**
    b. **Callback URL**
    c. **Webhook URL**
5. Scroll to the bottom of the page and click **Save**.
6. In a separate tab, navigate to the address you entered as the new homepage URL for your GitHub application.
7. Choose GitHub as the sign-in method by clicking the **SIGN IN** button.
8. In the popup window, authorize your custom GitHub application as requested.

You should be immediately redirected to Red Hat Developer Hub (RHDH). Any developer who downloads your GitHub application can also authenticate by using that application, and by running the second PipelineRun generated in the third procedure. In RHDH, developers can then leverage the automated, customizable, and secure CI/CD functionality of Red Hat Trusted Application Pipeline.

Revised on 2024-06-17 13:04:18 UTC