



Red Hat Trusted Application Pipeline 1.0

Customizing Red Hat Trusted Application Pipeline

Learn how to customize default software templates and build pipeline configurations.

Red Hat Trusted Application Pipeline 1.0 Customizing Red Hat Trusted Application Pipeline

Learn how to customize default software templates and build pipeline configurations.

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for cluster administrators on customizing RHTAP's software templates and build pipeline configurations to better align with the unique requirements of on-prem environments. By customizing these elements, administrators can ensure that development workflows are streamlined, security practices are seamlessly integrated, and developers can focus their efforts on innovation rather than the intricacies of security compliance and infrastructure nuances.

Table of Contents

PREFACE	3
CHAPTER 1. CUSTOMIZING SAMPLE SOFTWARE TEMPLATES	4
CHAPTER 2. CUSTOMIZING SAMPLE PIPELINES	8
Customizing the sample templates repository to update pac URLs*	8
Customizing the sample pipelines repository to your workflow	8
CHAPTER 3. CONFIGURING GITLAB WEBHOOKS FOR AUTOMATED PIPELINE TRIGGERS	10

PREFACE

RHTAP empowers teams with its ready-to-use software templates and build pipeline configurations, designed to seamlessly integrate security practices into your development processes. These tools not only alleviate the burden of security considerations for developers but also enhance focus on innovation.

Cluster administrators play a pivotal role in tailoring these resources to fit the unique requirements of their on-prem environments, including:

- Customizing software templates to meet specific organizational needs
- Modifying build pipeline configurations to align with project goals
- Configuring GitLab Webhooks for automated pipeline triggers

Such customizations streamline development workflows, addressing common concerns around pipelines, vulnerabilities, and policy compliance, thereby letting developers prioritize coding.

CHAPTER 1. CUSTOMIZING SAMPLE SOFTWARE TEMPLATES

Learn how to customize ready-to-use software templates for your on-prem environment. Cluster administrators have full control over this process, including modifying metadata and specifications.

Prerequisites

- You have used the forked repository URL from [tssc-sample-templates](#) during the RHTAP install process.

Procedure

1. Clone your forked repository, and then open it in your preferred text editor, such as Visual Studio Code.
2. Locate the **properties** file within your project directory. This file stores the default values that can customize. Open it for editing and update the following key-value pairs according to your environment.

Key	Description
export GITHUB_DEFAULT_HOST	Set this to your on-prem GitHub host fully qualified domain name. That is, the URL without the HTTP protocol and without the .git extension. For example github-github.apps.cluster-ljg9z.sandbox219.opentlc.com. Default is github.com .
export GITLAB_DEFAULT_HOST	Set this to your on-prem GitLab host fully qualified domain name. That is, the URL without the HTTP protocol and without the .git extension. For example gitlab-gitlab.apps.cluster-ljg9z.sandbox219.opentlc.com. Default is gitlab.com .
export QUAY_DEFAULT_HOST	The default Quay URL correspond to your specific on-prem image registry URL without the HTTP protocol. For example, quay-tv2pb.apps.cluster-tv2pb.sandbox1194.opentlc.com. The default quay host is quay.io .


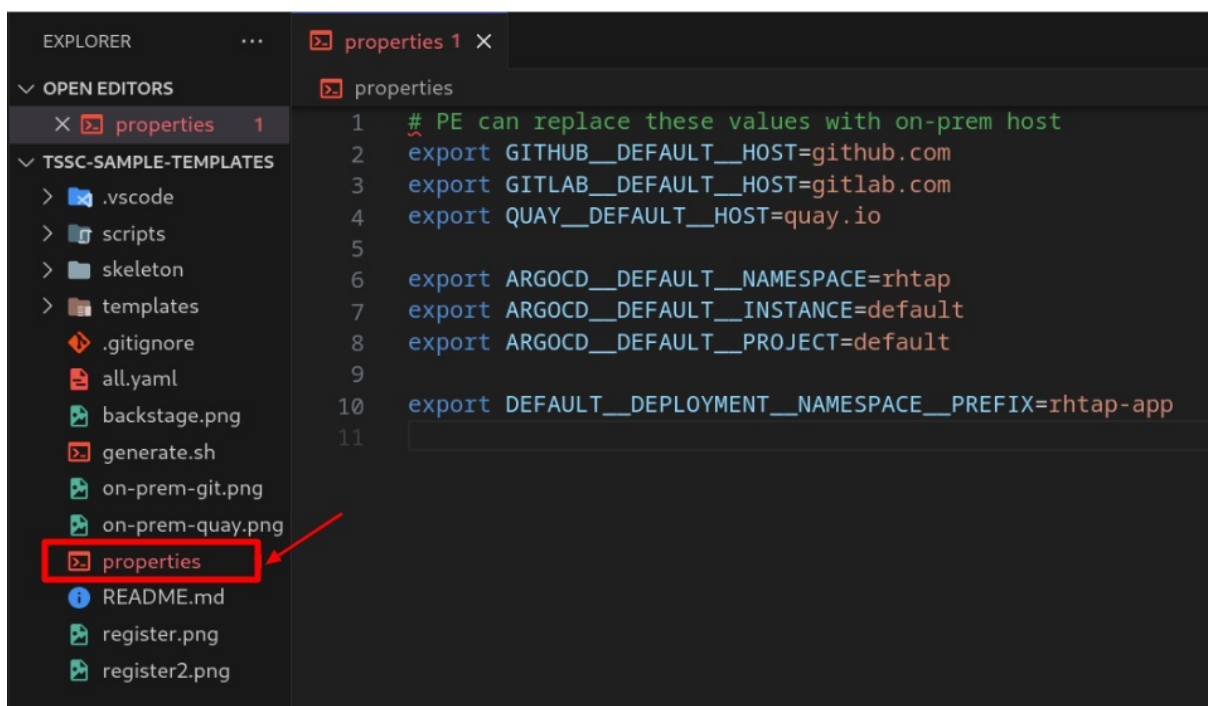
Key	Description
export DEFAULT_DEPLOYMENT_NAMESPACE_PREFIX X	<p>The namespace prefix for deployments within RHTAP. Default is rhtap-app.</p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 1;"> <p>NOTE</p> <p>Update this if you have modified the default trusted-application-pipeline: namespace during the RHTAP installation process.</p> </div> </div>

Figure 1.1. The properties file



```

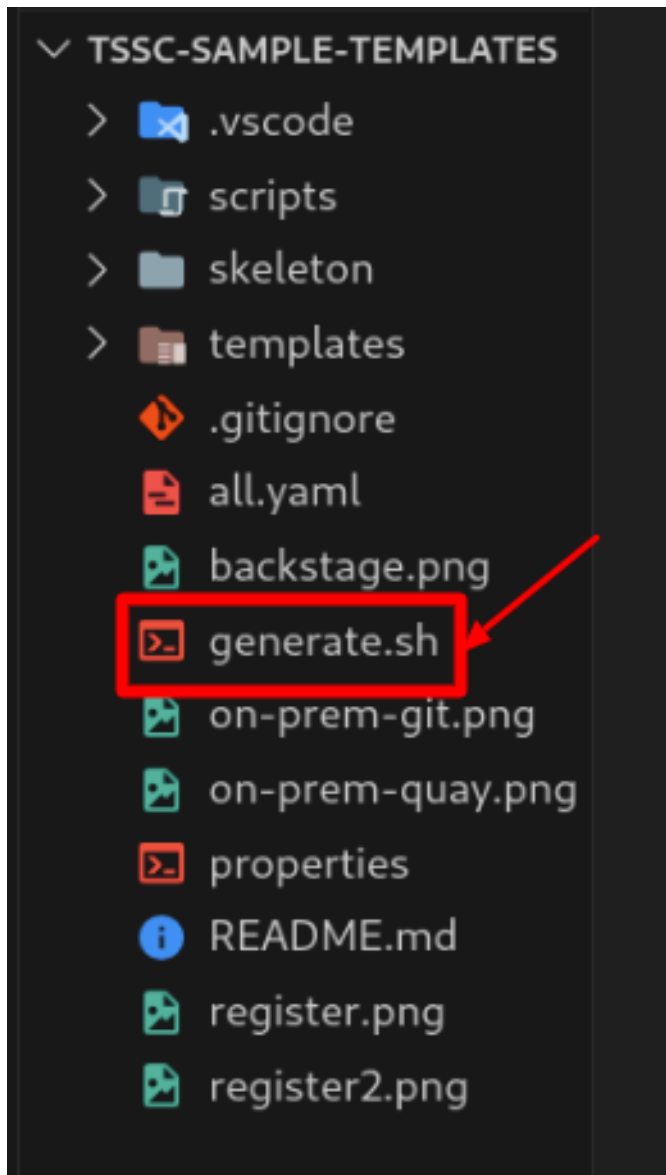
1  # PE can replace these values with on-prem host
2  export GITHUB_DEFAULT_HOST=github.com
3  export GITLAB_DEFAULT_HOST=gitlab.com
4  export QUAY_DEFAULT_HOST=quay.io
5
6  export ARGOCD_DEFAULT_NAMESPACE=rhtap
7  export ARGOCD_DEFAULT_INSTANCE=default
8  export ARGOCD_DEFAULT_PROJECT=default
9
10 export DEFAULT_DEPLOYMENT_NAMESPACE_PREFIX=rhtap-app
11

```

- Run the **generate.sh** script in your terminal. This action adjusts the software templates, replacing default host values with your specified inputs.

```
./generate.sh
```

Figure 1.2. The generate.sh script



4. Commit and push the changes to your repository. This automatically updates the template in RHDH. Alternatively, you can import and refresh a single or all customized templates directly in RHDH.
 - a. Go to your forked sample template repository on your Git provider.
 - b. For a single template, from the **templates** directory, select select **template.yaml**. Copy its URL from the browser address bar. For example, <https://github.com/<username>/tssc-sample-templates/blob/main/templates/devfile-sample-code-with-quarkus-dance/template.yaml>. Otherwise, for all the templates, select **all.yaml** and copy its URL from the browser address bar. For example, <https://github.com/<username>/tssc-sample-templates/blob/main/all.yaml>.
 - c. Switch back to RHDH platform.
 - d. Select **Create > Register Existing Component**.
 - e. In the **Select URL** field, paste the appropriate URL that you copied in Step 4b.
 - f. Select **Analyze** and then select **Import** to update the templates in RHDH.

Verification

- Consider creating an application to explore the impact of your template customization.

Additional resources

- To customize pipelines, see [Customizing sample pipeline templates](#)

CHAPTER 2. CUSTOMIZING SAMPLE PIPELINES

Learn how to update Pipeline as Code (**pac**) URLs within the sample templates repository and to customize the sample pipelines repository to your workflow. By customizing **pac** URLs, organizations can leverage specific pipelines tailored to their needs.

Prerequisites

- You have already forked and cloned the following templates locally:
 - [Sample pipelines](#)
 - [Sample templates](#)

Customizing the sample templates repository to update **pac** URLs*

Procedure

1. Access forked sample pipelines repository URL:
 - a. Open your forked sample pipelines repository.
 - b. Copy the complete URL from the address bar. For example, <https://github.com/<username>/tssc-sample-pipelines>.
2. Update **pac** URLs in the sample templates repository
 - a. Navigate to your local cloned sample templates repository using your terminal.
 - b. Run the following command, replacing `{fork_url}` with the copied URL from step 1 and `{branch_name}` with your desired branch name (for example, `main`):

```
./scripts/update-tekton-definition {fork_url} {branch_name}
```

```
# For example, ./scripts/update-tekton-definition https://github.com/<username>/tssc-sample-pipelines main
```

3. Review, commit, and push changes:
 - a. Review the updated files within your sample templates repository.
 - b. Commit the changes with appropriate message.
 - c. Push the committed changes to your forked repository.

Customizing the sample pipelines repository to your workflow

The sample pipelines repository provides a foundation upon which you can build your organization's specific CI/CD workflows. The sample pipelines repository includes several key pipeline templates in the **pac** directory:

- **gitops-repo**: This directory holds the pipeline definitions for validating pull requests within your GitOps repository. It triggers the **gitops-pull-request** pipeline, located in the **pipelines** directory, validating that image updates comply with organizational standards. This setup is crucial for promotion workflows, where an application's deployment state is advanced

sequentially through environments, such as from development to staging or from staging to production. For more information about pipeline definitions in **gitops-repo**, refer [Gitops Pipelines](#).

- **pipelines:** This directory houses the implementations of build and validation pipelines that are referenced by the event handlers in both the **gitops-repo** and **source-repo**. By examining the contents of this directory, you can understand the specific actions performed by the pipelines, including how they contribute to the secure promotion and deployment of applications.
- **source-repo:** This directory focuses on Dockerfile-based secure supply chain software builds. It includes pipeline definitions for cloning the source, generating and signing artifacts (such as **.sig** for image signature, **.att** for attestation, and **.sbom** for Software Bill of Materials), and pushing these to the user's image registry. For more information about pipeline definitions in **source-repo**, refer [Shared Git resolver model for shared pipeline and tasks](#).
- **tasks:** This directory houses a collection of tasks that can be added or modified, aligning with organizational needs. For example, Advanced Cluster Security (ACS) tasks can be substituted with alternative checks, or entirely new tasks can be integrated into the pipeline to enhance its functionality and compliance.

Verification

- Consider creating an application to explore the impact of your template and pipeline customization.

Additional resources

- To customize templates, see [Customizing sample software templates](#)
- For information on Pipeline as code, refer [About Pipelines as Code](#).

CHAPTER 3. CONFIGURING GITLAB WEBHOOKS FOR AUTOMATED PIPELINE TRIGGERS

Learn how to set up webhooks and secrets in GitLab to automatically trigger pipeline run in RHDH upon code updates.

Prerequisites

- You have an existing GitLab project.
- You have [administrator privileges](#) on OpenShift web console.

Procedure

1. **Retrieve Webhook URL and Secret Token:**
 - a. Log in to the [OpenShift web console](#) with Administrator privileges.
 - b. Navigate to the **rhtap** project, expand **Pipelines**, and then select **PipelineRuns**.
 - c. Locate the **rhtap-pe-info-<>** pipeline run, and then select the **Logs** tab.



NOTE

These logs contain the webhook URL and secret token required for GitLab configuration.

2. **Configure Webhook in GitLab:**
 - a. Within your GitLab repository, navigate to **Settings > Webhooks**.
 - b. In the **URL** field, enter the webhook URL copied from Step 1.
 - c. In the **Secret Token** field, enter the secret token copied from Step 1.
 - d. In the **Trigger** section:
 - i. Select **Push events**.
 - ii. Select **Merge request events**.
 - e. Click **Add Webhook**.

Verification

1. Push your code changes to the GitLab repository.
2. Navigate to the **CI** tab in RHDH.
3. Verify that a pipeline run is triggered for your code push.

Revised on 2024-07-01 13:31:59 UTC