



Red Hat Developer Hub 1.8

Streamline software development and management in Red Hat Developer Hub

Automating the development lifecycle, improving code quality, monitoring deployments, and managing services by using tools and plugins in Red Hat Developer Hub (RHDH)

Red Hat Developer Hub 1.8 Streamline software development and management in Red Hat Developer Hub

Automating the development lifecycle, improving code quality, monitoring deployments, and managing services by using tools and plugins in Red Hat Developer Hub (RHDH)

Legal Notice

Copyright © Red Hat.

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, 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 Software Collections 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

As a developer, you can learn how to use the platform integrated tools and services to create component catalogs, enforce code standards, manage security, and automate the software development and management workflow in Red Hat Developer Hub (RHDH).

Table of Contents

CHAPTER 1. IMPORTING YOUR TEAM’S CODEBASE FROM GIT	3
1.1. ENABLING AND AUTHORIZING BULK IMPORT CAPABILITIES IN RED HAT DEVELOPER HUB	3
1.2. IMPORTING MULTIPLE GITHUB REPOSITORIES	4
1.3. MANAGING THE ADDED GIT REPOSITORIES	5
1.4. MONITORING BULK IMPORT ACTIONS USING AUDIT LOGS	6
CHAPTER 2. CENTRALIZING YOUR SOFTWARE COMPONENTS IN THE RED HAT DEVELOPER HUB CATALOG FOR EASIER ACCESS	8
2.1. ADDING NEW COMPONENTS TO YOUR RED HAT DEVELOPER HUB INSTANCE TO EXPAND YOUR CATALOG	8
2.1.1. Creating new components in your Red Hat Developer Hub instance	8
2.1.2. Registering components manually in your RHDH instance	9
2.2. UPDATING EXISTING COMPONENT IN YOUR RED HAT DEVELOPER HUB CATALOG	10
2.3. FINDING THE RIGHT COMPONENTS QUICKLY IN THE RED HAT DEVELOPER HUB CATALOG BY KIND	10
2.4. FINDING THE RIGHT COMPONENT QUICKLY IN THE RED HAT DEVELOPER HUB CATALOG BY USING THE FILTER FIELD	10
2.5. REVIEWING THE YAML CONFIGURATION OF YOUR RED HAT DEVELOPER HUB SOFTWARE CATALOG	11
2.6. STARRING KEY COMPONENTS IN THE SOFTWARE CATALOG	11
CHAPTER 3. IMPORTING AND USING AN EXISTING SOFTWARE TEMPLATE FOR FASTER DEVELOPMENT .	12
3.1. CREATING A SOFTWARE TEMPLATE BY USING THE TEMPLATE EDITOR	12
3.2. CREATING A SOFTWARE TEMPLATE AS A YAML FILE	13
3.3. CREATING A NEW SOFTWARE COMPONENT USING SOFTWARE TEMPLATES	15
3.4. SEARCHING AND FILTERING SOFTWARE TEMPLATES IN YOUR RED HAT DEVELOPER HUB INSTANCE	17
3.5. IMPORTING AN EXISTING SOFTWARE TEMPLATE TO RED HAT DEVELOPER HUB	17

CHAPTER 1. IMPORTING YOUR TEAM'S CODEBASE FROM GIT



IMPORTANT

These features are for Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend using them for 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 on Red Hat Technology Preview features, see [Technology Preview Features Scope](#).

Red Hat Developer Hub can automate GitHub repositories onboarding and track their import status.

1.1. ENABLING AND AUTHORIZING BULK IMPORT CAPABILITIES IN RED HAT DEVELOPER HUB

You can enable the Bulk Import feature for users and give them the necessary permissions to access it. This feature is available for GitHub repositories and GitLab projects.

Prerequisites

- For GitHub only: You have [enabled GitHub repository discovery](#).

Procedure

- The Bulk Import plugins are installed but disabled by default. To enable the **./dynamic-plugins/dist/red-hat-developer-hub-backstage-plugin-bulk-import-backend-dynamic** and **./dynamic-plugins/dist/red-hat-developer-hub-backstage-plugin-bulk-import** plugins, edit your **dynamic-plugins.yaml** with the following content:

```
plugins:
  - package: ./dynamic-plugins/dist/red-hat-developer-hub-backstage-plugin-bulk-import-backend-dynamic
    disabled: false
  - package: ./dynamic-plugins/dist/red-hat-developer-hub-backstage-plugin-bulk-import
    disabled: false
```

See [Installing and viewing plugins in Red Hat Developer Hub](#).

- Configure the required **bulk.import** RBAC permission for the users who are not administrators as shown in the following code:

rbac-policy.csv fragment

```
p, role:default/bulk-import, bulk.import, use, allow
g, user:default/<your_user>, role:default/bulk-import
```

Note that only Developer Hub administrators or users with the **bulk.import** permission can use the Bulk Import feature. See [Permission policies in Red Hat Developer Hub](#).

Verification

- The sidebar displays a **Bulk Import** option.
- The **Bulk Import** page shows a list of added GitHub repositories and GitLab projects.

1.2. IMPORTING MULTIPLE GITHUB REPOSITORIES

In Red Hat Developer Hub, you can select your GitHub repositories and automate their onboarding to the Developer Hub catalog.

Prerequisites

- You have [enabled the Bulk Import feature and gave access to it](#).

Procedure

1. Click **Bulk Import** in the left sidebar.
2. Click the **Add** button in the top-right corner to see the list of all repositories accessible from the configured GitHub integrations.
 - a. From the **Repositories** view, you can select any repository, or search for any accessible repositories. For each repository selected, a **catalog-info.yaml** is generated.
 - b. From the **Organizations** view, you can select any organization by clicking **Select** in the third column. This option allows you to select one or more repositories from the selected organization.
3. Click **Preview file** to view or edit the details of the pull request for each repository.
 - a. Review the pull request description and the **catalog-info.yaml** file content.
 - b. Optional: when the repository has a **.github/CODEOWNERS** file, you can select the **Use CODEOWNERS file as Entity Owner** checkbox to use it, rather than having the **content-info.yaml** contain a specific entity owner.
 - c. Click **Save**.
4. Click **Create pull requests**. At this point, a set of dry-run checks runs against the selected repositories to ensure they meet the requirements for import, such as:
 - a. Verifying that there is no entity in the Developer Hub catalog with the name specified in the repository **catalog-info.yaml**
 - b. Verifying that the repository is not empty
 - c. Verifying that the repository contains a **.github/CODEOWNERS** file if the **Use CODEOWNERS file as Entity Owner** checkbox is selected for that repository
 - If any errors occur, the pull requests are not created, and you see a *Failed to create PR* error message detailing the issues. To view more details about the reasons, click **Edit**.
 - If there are no errors, the pull requests are created, and you are redirected to the list of added repositories.
5. Review and merge each pull request that creates a **catalog-info.yml** file.

Verification

- The **Added entities** list displays the repositories you imported, each with an appropriate status: either *Waiting for approval* or *Added*.
- For each *Waiting for approval* import job listed, there is a corresponding pull request adding the **catalog-info.yaml** file in the corresponding repository.

1.3. MANAGING THE ADDED GIT REPOSITORIES

You can oversee and manage the Git repositories that are imported to the Developer Hub.

Prerequisites

- You have [imported GitHub repositories](#).

Procedure

- Click **Bulk Import** in the left sidebar to display all the current GitHub repositories and GitLab projects that are being tracked as Import jobs, along with their status.

Added

The Git repository is added to the Developer Hub catalog after the import pull request is merged or if the Git repository already contained a **catalog-info.yaml** file during the bulk import. It can take a few minutes for the entities to be available in the catalog.

Waiting for approval

There is an open pull request or merge request adding a **catalog-info.yaml** file to the GitHub repository or GitLab project. You can:

- Click **pencil** icon on the right to see details about the pull request or merge request. You can use the detailed view to edit the request content right from Developer Hub.
- Delete the Import job, this action closes the import pull request or merge request as well.
- To transition the Import job to the *Added* state, merge the import pull request or merge request from the Git repository.

Empty

Developer Hub is unable to determine the import job status because the Git repository is imported from other sources but does not have a **catalog-info.yaml** file and lacks any import pull or merge request adding it.



NOTE

- After an import pull request or merge request is merged, the import status is marked as *Added* in the list of **Added entities**, but it might take a few seconds for the corresponding entities to appear in the Developer Hub Catalog.
- A location added through other sources (for example, statically in an **app-config.yaml** file, dynamically when [enabling GitHub discovery](#), or registered manually using the "Register an existing component" page) might show up in the Bulk Import list of Added Repositories if the following conditions are met:
 - The location URL points to a **catalog-info.yaml** file at the root of the Git repository default branch.
 - For GitHub only: The target repository is accessible from the configured GitHub integrations.

1.4. MONITORING BULK IMPORT ACTIONS USING AUDIT LOGS

The Bulk Import backend plugin adds the following events to the Developer Hub audit logs. See [Audit logs in Red Hat Developer Hub](#) for more information on how to configure and view audit logs.

Bulk Import Events:

BulkImportUnknownEndpoint

Tracks requests to unknown endpoints.

BulkImportPing

Tracks **GET** requests to the **/ping** endpoint, which allows us to make sure the bulk import backend is up and running.

BulkImportFindAllOrganizations

Tracks **GET** requests to the **/organizations** endpoint, which returns the list of organizations accessible from all configured GitHub Integrations.

BulkImportFindRepositoriesByOrganization

Tracks **GET** requests to the **/organizations/:orgName/repositories** endpoint, which returns the list of repositories for the specified organization (accessible from any of the configured GitHub Integrations).

BulkImportFindAllRepositories

Tracks GET requests to the **/repositories** endpoint, which returns the list of repositories accessible from all configured GitHub Integrations.

BulkImportFindAllImports

Tracks **GET** requests to the **/imports** endpoint, which returns the list of existing import jobs along with their statuses.

BulkImportCreateImportJobs

Tracks **POST** requests to the **/imports** endpoint, which allows to submit requests to bulk-import one or many repositories into the Developer Hub catalog, by eventually creating import pull requests in the target repositories.

BulkImportFindImportStatusByRepo

Tracks **GET** requests to the **/import/by-repo** endpoint, which fetches details about the import job for the specified repository.

BulkImportDeleteImportByRepo

Tracks **DELETE** requests to the **/import/by-repo** endpoint, which deletes any existing import job for the specified repository, by closing any open import pull request that could have been created.

Example bulk import audit logs

```
{
  "actor": {
    "actorId": "user:default/myuser",
    "hostname": "localhost",
    "ip": "::1",
    "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/128.0.0.0 Safari/537.36"
  },
  "eventName": "BulkImportFindAllOrganizations",
  "isAuditLog": true,
  "level": "info",
  "message": "'get /organizations' endpoint hit by user:default/myuser",
  "meta": {},
  "plugin": "bulk-import",
  "request": {
    "body": {},
    "method": "GET",
    "params": {},
    "query": {
      "pagePerIntegration": "1",
      "sizePerIntegration": "5"
    }
  },
  "url": "/api/bulk-import/organizations?pagePerIntegration=1&sizePerIntegration=5"
},
"response": {
  "status": 200
},
"service": "backstage",
"stage": "completion",
"status": "succeeded",
"timestamp": "2024-08-26 16:41:02"
}
```

CHAPTER 2. CENTRALIZING YOUR SOFTWARE COMPONENTS IN THE RED HAT DEVELOPER HUB CATALOG FOR EASIER ACCESS

The Red Hat Developer Hub Software Catalog is a centralized system that gives you visibility into all the software across your ecosystem, including services, websites, libraries, and data pipelines. You can use it to view ownership details and metadata for each component in one place.

The metadata for the components in your Software Catalog is stored as YAML files that live alongside your code in your version control system. The version control repositories can include one or many metadata files. Software Catalog organizes items as entities, which include Components, Resources, and APIs, and other related types. Each entity includes associated metadata such as its owner, type, and other relevant details.

By storing metadata in YAML files alongside the code, you allow Red Hat Developer Hub to process and display this information through a clear, visual interface. With the Software Catalog, you can manage and maintain your software, stay aware of all software available in your ecosystem, and take ownership of your services and tools.

The **Overview** page for a component provides key information such as links to the source code, documentation, dependencies, and ownership details. You can customize this page with plugins to suit specific needs.

2.1. ADDING NEW COMPONENTS TO YOUR RED HAT DEVELOPER HUB INSTANCE TO EXPAND YOUR CATALOG

Prerequisites

- You have installed and configured the Red Hat Developer Hub instance.
- You have the required permissions. See [Authorization in Red Hat Developer Hub](#) .

Procedure

You can add components to your RHDH instance using the following methods:

- Register components manually using the GUI or by using your **app-config.yaml** with the required permissions.
- Create new components by using Software Templates.
- Use the bulk import plugin with the required permissions. For more information, see [Bulk importing GitHub repositories](#).

2.1.1. Creating new components in your Red Hat Developer Hub instance

You can create new components in the Software Catalog in your RHDH instance. Red Hat Developer Hub automatically registers all components that developers or platform engineers create using Templates in the Software Catalog.

Prerequisites

- You have installed and configured the Red Hat Developer Hub instance.

- You have the required permissions. See [Authorization in Red Hat Developer Hub](#) .

Procedure

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. On the **Catalog** page, click **Self-service**.

2.1.2. Registering components manually in your RHDH instance

To manually register components in your RHDH instance, create a **catalog-info.yaml** file and register it with your Red Hat Developer Hub instance. The **catalog-info.yaml** file contains the metadata you wish to register for your software component.

Prerequisites

- You have installed and configured the Red Hat Developer Hub instance.
- You have the required permissions. See [Authorization in Red Hat Developer Hub](#) .

Procedure

1. In the root directory of your software project, create a file named **catalog-info.yaml**.

```
apiVersion: backstage.io/v1alpha1
kind: Component
metadata:
  name: __<your_software_component>__
  description: __<software_component_brief_description>__
  tags:
    - example
    - service
  annotations:
    github.com/project-slug: __<repo_link_of_your_component_to_register>__
spec:
  type: __<your_service>__
  owner: __<your_team_name>__
  lifecycle: __<your_lifecycle>__
```

2. Commit the **catalog-info.yaml** file to the root of your project source code repository.
3. In your Red Hat Developer Hub navigation menu, go to **Catalog > Self-service**.
4. On the **Self-service** page, click **Register Existing Component**.
5. On the **Register an existing component** page, enter the full URL of the **catalog-info.yaml** file in your repository. For example: [Artist lookup component](#).
6. Complete the wizard instructions.

Verification

- Your software component is listed in the Software Catalog. You can view its details and ensure all the metadata is accurate.

2.2. UPDATING EXISTING COMPONENT IN YOUR RED HAT DEVELOPER HUB CATALOG

You can update components in the Software Catalog in your Red Hat Developer Hub instance.

Prerequisites

- You have installed and configured the Red Hat Developer Hub instance.
- You have the required permissions. See [Authorization in Red Hat Developer Hub](#).

Procedure

To update components in the Software Catalog in your Red Hat Developer Hub instance, complete the following steps:

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. Find the software component that you want to edit, under **Actions**, click the **Edit** icon.



NOTE

This action redirects you to the YAML file on GitHub.

3. On your remote repository UI, update your YAML file.



NOTE

After you merge your changes, the updated metadata in the Software Catalog appears after some time.

2.3. FINDING THE RIGHT COMPONENTS QUICKLY IN THE RED HAT DEVELOPER HUB CATALOG BY KIND

Procedure

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. On the **Catalog** page, click the **Kind** drop-down list.
3. Select the type of **Kind** you want to filter.



NOTE

The available filter dropdowns vary based on the **Kind** you select, displaying options relevant to that specific entity type.

2.4. FINDING THE RIGHT COMPONENT QUICKLY IN THE RED HAT DEVELOPER HUB CATALOG BY USING THE FILTER FIELD

Procedure

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. In the **Search** box, enter the text you want to use to filter the components.

2.5. REVIEWING THE YAML CONFIGURATION OF YOUR RED HAT DEVELOPER HUB SOFTWARE CATALOG

You can view the Software Catalog YAML file in your Red Hat Developer Hub instance. The YAML file displays the metadata for the components in your Software Catalog.

Procedure

To view the Software Catalog YAML file in your Red Hat Developer Hub instance, complete the following steps:

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. Find the software component that you want to view, under **Actions**, click the **View** icon.



NOTE

These steps redirect you to the YAML file on your remote repository.

2.6. STARRING KEY COMPONENTS IN THE SOFTWARE CATALOG

You can use the **Add to favorites** icon to add the software catalogs that you visit regularly to the **Starred** category.

Procedure

To quickly access the Software Catalogs that you visit regularly, complete the following steps:

1. In your Red Hat Developer Hub navigation menu, click **Catalog**.
2. Find the software component that you want to add as a favorite, under **Actions**, click the **Add to favorites** icon.

Verification

- The starred component is listed under **Your Starred Entities** on your **Home** page.

CHAPTER 3. IMPORTING AND USING AN EXISTING SOFTWARE TEMPLATE FOR FASTER DEVELOPMENT

You can configure Software Templates to create software components and publish these components to Git repositories. Once the components are published to Git repositories, register these components in the Software Catalog.

A template is a form composed of different UI fields that is defined in a YAML file. Software Templates include *actions*, which are steps that are executed in sequential order and can be executed conditionally.

3.1. CREATING A SOFTWARE TEMPLATE BY USING THE TEMPLATE EDITOR

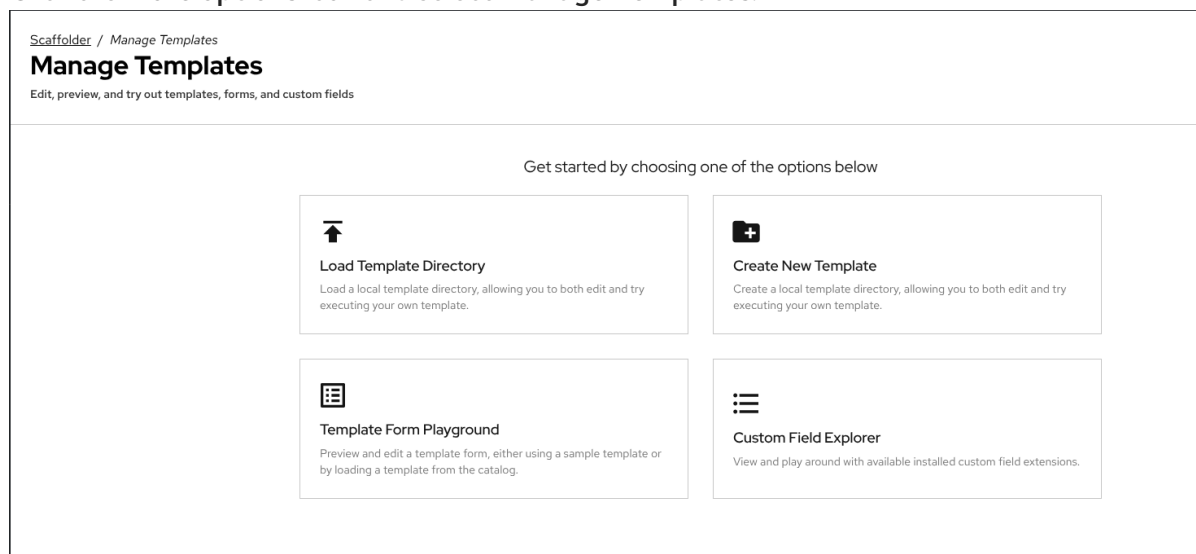
Use the Red Hat Developer Hub Template Editor to create a Software Template.

Alternately, you can use the Template Editor to do any of the following actions:

- **File > Open template directory**
- **File > Create template directory**
- **File > Close template editor**
- Use the **Custom Fields Explorer** button to test custom fields in your **templates.yaml** file
- **View Installed Actions Documentation**

Procedure

1. In your Red Hat Developer Hub navigation menu, click **Catalog > Self-service**. Alternatively, to go to **Self-service** page, in your header menu, click the (+) icon.
2. Click the **More options** icon and select **Manage Templates**.





NOTE

- The following options on the **Managed Templates** page do not create a software component in your Red Hat Developer Hub instance:
 - **Template Form Playground** Use to create and test the **templates.yaml** file
 - **Custom Field Explorer**: Use to test custom fields

3. On the **Managed Templates** page, select any of the following options:

- **Load Template Directory**: Use to load an existing **templates.yaml** file
 - In your local file manager, navigate to the folder where your **templates.yaml** file is stored and click **Select**.
- **Create New Template**: Use to create a **templates.yaml** file
 - a. In your local file manager, navigate to the folder where you want the Template Editor to create a **templates.yaml** file and click **Select**.
 - b. On the **Template Editor** page, select the **templates.yaml** file.
 - c. (Optional) You can preview and test your template specifications.
 - i. On the **Fill in some steps** tab, enter text into the required fields and click **Next**.
 - ii. On the **Repository Location** tab, enter text into the required fields and click **Review**.
 - iii. Modify the YAML definition for the parameters of your template. For more information about these parameters, see [Section 3.2, "Creating a Software Template as a YAML file"](#).
 - iv. Review the information for accuracy, then click **Create**.
 - d. After the Software Template is created, [import your Software Template in your RHDH instance](#).

Verification

1. Click the **Catalog** tab in the navigation panel.
2. In the **Kind** drop-down menu, select **Template**.
3. Confirm that your template is shown in the list of existing templates.

Next step

- [Import your Software Template in your RHDH instance](#) .

3.2. CREATING A SOFTWARE TEMPLATE AS A YAML FILE

You can create a Software Template by defining a **Template** object as a YAML file.

The **Template** object describes the Software Template and its metadata. It also contains required input variables and a list of actions that are executed by the scaffolding service.

Template object example

```
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  name: template-name ❶
  title: Example template ❷
  description: An example template for v1beta3 scaffolder. ❸
spec:
  owner: backstage/techdocs-core ❹
  type: service ❺
  parameters: ❻
    - title: Fill in some steps
      required:
        - name
      properties:
        name:
          title: Name
          type: string
          description: Unique name of the component
        owner:
          title: Owner
          type: string
          description: Owner of the component
    - title: Choose a location
      required:
        - repoUrl
      properties:
        repoUrl:
          title: Repository Location
          type: string
  steps: ❼
    - id: fetch-base
      name: Fetch Base
      action: fetch:template
      # ...
  output: ❽
    links:
      - title: Repository ❾
        url: ${ steps['publish'].output.remoteUrl }
      - title: Open in catalog ❿
        icon: catalog
        entityRef: ${ steps['register'].output.entityRef }
  # ...
```

- ❶ Specify a name for the Software Template.
- ❷ Specify a title for the Software Template. This is the title that is visible on the Software Template tile in the **Self-service** view.
- ❸ Specify a description for the Software Template. This is the description that is visible on the Software Template tile in the **Self-service** view.

Software Template tile in the **Self-service** view.

- 4 Specify the ownership of the Software Template. The **owner** field provides information about who is responsible for maintaining or overseeing the Software Template within the system or organization. In the provided example, the **owner** field is set to **backstage/techdocs-core**. This means that this Software Template belongs to the **techdocs-core** project in the **backstage** namespace.
- 5 Specify the component type. Any string value is accepted for this required field, but your organization should establish a proper taxonomy for these. Red Hat Developer Hub instances may read this field and behave differently depending on its value. For example, a **website** type component may present tooling in the Red Hat Developer Hub interface that is specific to just websites.

The following values are common for this field:

service

A backend service, typically exposing an API.

website

A website.

library

A software library, such as an npm module or a Java library.

- 6 Use the **parameters** section to specify parameters for user input that are shown in a form view when a user creates a component by using the Software Template in the Red Hat Developer Hub console. Each **parameters** subsection, defined by a title and properties, creates a new form page with that definition.
- 7 Use the **steps** section to specify steps that are executed in the backend. These steps must be defined by using a unique step ID, a name, and an action. You can view actions that are available on your Red Hat Developer Hub instance by visiting the URL **`https://<rhdh_url>/create/actions`**.
- 8 Use the **output** section to specify the structure of output data that is created when the template is used. The **output** section, particularly the **links** subsection, provides valuable references and URLs that users can utilize to access and interact with components that are created from the template.
- 9 Provides a reference or URL to the repository associated with the generated component.
- 10 Provides a reference or URL that allows users to open the generated component in a catalog or directory where various components are listed.

3.3. CREATING A NEW SOFTWARE COMPONENT USING SOFTWARE TEMPLATES

You can create a new software component using the standard Software Templates that the platform engineers have created. The scaffolding process runs in your Red Hat Developer Hub instance.

Procedure

1. In your Red Hat Developer Hub navigation menu, click **Catalog** > **Self-service**.
2. On the **Self-service** page, click **Choose** on the **Templates** tile to initiate the scaffolding process for a template.

- Follow the wizard instructions as you enter the required details. You can choose parameters from a set of pre-defined options.

- Optional: In the **Deployment Information** step, you have an option to **Create Workbench for OpenShift AI**.



NOTE

This step is available only for a few templates.

- In the **Review** step, verify the parameters you have entered and click **Create**.



NOTE

- You can click **Cancel** to abort the software component creation during the template running step only if the current step supports the abort. The abort signal is then sent to a task and none of the following steps are executed.
- During the creation of the software component, click **Show Logs** to view the log information.

Verification

- If your software component is not created successfully, you can review the logs on the error page. To return to the **Self-service** page with the same template form and your previously entered values, click **Start Over**.

The screenshot shows the 'Run of Add ArgoCD to an existing project' task page. The task ID is 'Task 85e2acf5-bf18-4424-a5cf-99e2835d34b9'. A red error banner at the top states: 'Error: Template action with ID 'publish:github' is not registered. See <https://backstage.io/docs/features/software-templates/builtin-actions/> on how to add a new action module.' Below the error, a progress bar shows five steps: 'Generating the Catalog Info Component' (2 seconds, green check), 'Generating the Manifests Component' (1 second, green check), 'Publishing to the Source Code Repository' (0 seconds, red circle), 'Registering the Catalog Info Component' (white circle), and 'Create the ArgoCD Resources' (white circle). At the bottom, there are buttons for 'Cancel', 'Hide Logs', and 'Start Over'. A log section at the bottom shows timestamps and messages for writing manifest files.

- If your Software Template is created successfully, a success page similar to the example in the following image is displayed:

The screenshot shows the 'Run of quarkus-web-template' task page. The task ID is 'Task 37ece928-e213-423e-bc08-66c0336c7a43'. A green progress bar at the top indicates success. Below the progress bar, a timeline shows six steps, all with green checkmarks: 'Fetch Skeleton + Template' (1 second), 'Publish' (2 seconds), 'Register' (0 seconds), 'Generating Deployment Resources' (1 second), 'Publishing to Resource Repository' (2 seconds), and 'Create ArgoCD Resources' (2 seconds). At the bottom, there are links for 'Source Code Repository' and 'Open Component in catalog'.

3.4. SEARCHING AND FILTERING SOFTWARE TEMPLATES IN YOUR RED HAT DEVELOPER HUB INSTANCE

You can search and filter for the Software Template that you want to use to create a new software component.

Procedure

1. In the Red Hat Developer Hub navigation menu, click **Catalog** > **Self-service**.
2. Type the name of the template you are looking for in the **Search** box.
 - If you are looking for templates in a certain category, you can use the **Categories** dropdown.

3.5. IMPORTING AN EXISTING SOFTWARE TEMPLATE TO RED HAT DEVELOPER HUB

You can add an existing Software Template to your Red Hat Developer Hub instance by using the Catalog Processor.

Prerequisites

- You have created a directory or repository that contains at least one template YAML file.
- Optional: To use a template stored in a GitHub repository, you have configured [Developer Hub integration with GitHub](#).

Procedure

- In the **app-config.yaml** configuration file, modify the **catalog.rules** section to include a rule for Software Templates, and configure the **catalog.locations** section to point to the Software Template that you want to add, as shown in the following example:

```
# ...
catalog:
  rules:
    - allow: [Template]
  locations:
    - type: url
      target: https://<repository_url>/template-name>.yaml
# ...
```

where:

catalog.rules.allow

Specify the **Template** rule to allow new Software Templates in the catalog.

catalog.locations.type

Specify the **url** type when importing templates from a repository (for example, GitHub or GitLab).

catalog.locations.target

Specify the URL for the template.

Verification

1. Click the **Catalog** tab in the navigation panel.
2. In the **Kind** drop-down menu, select **Template**.
3. Confirm that your template is shown in the list of existing templates.