Red Hat Developer Hub 1.8

TechDocs for Red Hat Developer Hub

Use the TechDocs plugin to read and manage your team's technical documentation in one place. Further enhance and customize your TechDocs experience with add-ons.

Last Updated: 2026-01-22

Red Hat Developer Hub 1.8 TechDocs for Red Hat Developer Hub

Use the TechDocs plugin to read and manage your team's technical documentation in one place. Further enhance and customize your TechDocs experience with add-ons.

Legal Notice

Abstract

Your organization can use the built-in TechDocs plugin for Red Hat Developer Hub (RHDH) to create, find, and use technical documentation in a central location and in a standardized way. Documentation files are stored alongside your code and rendered in the Docs tab. Use supported TechDocs add-ons, or create your own, to further enhance your documentation experience.

Table of Contents

CHAPTER 1. ABOUT TECHDOCS 3

CHAPTER 2. CONFIGURING TECHDOCS 4
2.1. CONFIGURING STORAGE FOR TECHDOCS FILES 4
2.1.1. Configuring Amazon S3 for file storage 4
2.1.2. Configuring OpenShift Data Foundation for file storage 5
2.1.2.1. Making object storage accessible to containers by using the Helm chart 6
2.1.2.1.1. Example TechDocs Plugin configuration for the Helm chart 7
2.1.2.2. Making object storage accessible to containers by using the Operator 8
2.1.2.2.1. Example TechDocs Plugin configuration for the Operator 8
2.2. CONFIGURING CI/CD TO GENERATE AND PUBLISH TECHDOCS SITES 9

CHAPTER 3. TECHDOCS ADD-ONS 11
3.1. INSTALLING AND CONFIGURING A TECHDOCS ADD-ON 11
3.1.1. Installing and configuring an external TechDocs add-on using the Operator 12
3.1.2. Installing and configuring an external TechDocs add-on using the Helm chart 14
3.1.3. Installing and configuring a third-party TechDocs add-on 15
3.2. REMOVING A TECHDOCS ADD-ON 17
3.2.1. Removing an external TechDocs add-on from your ConfigMap 17
3.2.2. Removing an external TechDocs add-on from your Helm chart 19
3.3. USING TECHDOCS ADD-ONS 20
3.3.1. Using the ReportIssue TechDocs add-on 20
3.3.2. Using the TextSize TechDocs add-on 20
3.3.3. Using the LightBox TechDocs add-on 21

CHAPTER 4. CREATING A TECHDOCS ADD-ON 22

CHAPTER 1. ABOUT TECHDOCS

The Red Hat Developer Hub instance comes with the TechDocs plugin preinstalled and enabled by default. Your organization can use the TechDocs plugin to create, find, and manage documentation in a central location and in a standardized way. You can also enhance your technical documentation experience with built-in TechDocs features and add-ons. For example:

Docs-like-code approach

Write your technical documentation in Markdown files that are stored inside your project repository along with your code.

Documentation site generation

Use MkDocs to create a full-featured, Markdown-based, static HTML site for your documentation that is rendered centrally in Developer Hub.

Documentation site metadata and integrations

See additional metadata about the documentation site alongside the static documentation, such as the date of the last update, the site owner, top contributors, open GitHub issues, Slack support channels, and Stack Overflow Enterprise tags.

Built-in navigation and search

Locate the information that you need within a document quickly and easily.

Add-ons

Make your documentation more functional and interactive with supported TechDocs add-ons. Some add-ons are preinstalled and enabled by default. To extend the default functionality, you can dynamically load external and third-party add-ons into your Red Hat Developer Hub instance. If you want to further customize your TechDocs experience, you can create add-ons that meet specific documentation needs for your organization.

CHAPTER 2. CONFIGURING TECHDOCS

The TechDocs plugin is preinstalled and enabled on a Developer Hub instance by default. You can disable or enable the TechDocs plugin, and change other parameters, by configuring the Red Hat Developer Hub Helm chart or the Red Hat Developer Hub Operator ConfigMap.

IMPORTANT

Red Hat Developer Hub includes a built-in TechDocs builder that generates static HTML documentation from your codebase. However, the default basic setup of the local builder is not intended for production.

You can use a CI/CD pipeline with the repository that has a dedicated job to generate docs for TechDocs. The generated static files are stored in OpenShift Data Foundation or in a cloud storage solution of your choice and published to a static HTML documentation site.

After you configure OpenShift Data Foundation to store the files that TechDocs generates, you can configure the TechDocs plugin to use the OpenShift Data Foundation for cloud storage.

2.1. CONFIGURING STORAGE FOR TECHDOCS FILES

The TechDocs publisher stores generated files in local storage or in cloud storage, such as AWS S3 or OpenShift Data Foundation.

2.1.1. Configuring Amazon S3 for file storage

> You can create a dedicated Amazon S3 bucket to store TechDocs sites. {product} uploads TechDocs to this bucket and serves them from the same location.

Prerequisites

- You are logged in to your AWS account as an administrator.

Procedure

1. On the AWS console, create an AWS S3 bucket.
   a. On the **Create bucket** page, enter a **Bucket name** and use the default selections for all other settings.
   b. Create an IAM policy to give authorized users permissions to generate and publish TechDocs for your organization.
   c. On the **Create policy > Specify permissions**page, in the **Policy editor**, enter the following JSON content:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TechDocsList",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
```

```
            "Resource": "arn:aws:s3:::_<bucket_name>_"
        },
        {
            "Sid": "TechDocsObjects",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::_<bucket_name>_/*"
        }
    ]
}
```

where

<bucket_name>

Specifies the name of your Amazon S3 bucket.

d. On the **Create policy > Specify permissions**page, enter a **Policy name**.

2. Assign the IAM policy to a new or existing user.
3. Generate a new access key and a new secret access key.

NOTE

You can use the newly created access keys to generate a TechDocs pipeline with GitHub Actions.

4. From the OpenShift Container Platform web console, click **Topology** > **Actions** > **Restart rollout** to restart the pod.

NOTE

You must restart the pod to apply the configuration changes.

Verification

1. Go to your Amazon S3 bucket to see a set of static site files in your **Objects** list.

2.1.2. Configuring OpenShift Data Foundation for file storage

You can configure OpenShift Data Foundation to store the files that TechDocs generates instead of relying on other cloud storage solutions.

OpenShift Data Foundation provides an **ObjectBucketClaim** custom resource (CR) that you can use to request an S3-compatible bucket backend. You must install the OpenShift Data Foundation Operator to use this feature.

+

NOTE

For air-gapped environments, using OpenShift Data Foundation is the recommended storage for TechDocs.

Prerequisites

- An OpenShift Container Platform administrator has installed the OpenShift Data Foundation Operator in Red Hat OpenShift Container Platform, created an OpenShift Data Foundation cluster and configured the **StorageSystem** schema. For more information, see Deploying OpenShift Data Foundation using Amazon Web Services.

Procedure

- Create an **ObjectBucketClaim** CR where the generated TechDocs files are stored. For example:

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <rhdh_bucket_claim_name>
spec:
  generateBucketName: <rhdh_bucket_claim_name>
  storageClassName: openshift-storage.noobaa.io
```

NOTE

Creating the Developer Hub **ObjectBucketClaim** CR automatically creates both the Developer Hub **ObjectBucketClaim** config map and secret. The config map and secret have the same name as the **ObjectBucketClaim** CR.

After you create the **ObjectBucketClaim** CR, you can use the information stored in the config map and secret to make the information accessible to the Developer Hub container as environment variables. Depending on the method that you used to install Developer Hub, you add the access information to either the Red Hat Developer Hub Helm chart or Operator configuration.

Additional resources

- OpenShift Data Foundation
- Object Bucket Claim

2.1.2.1. Making object storage accessible to containers by using the Helm chart

Creating a **ObjectBucketClaim** custom resource (CR) automatically generates both the Developer Hub **ObjectBucketClaim** config map and secret. The config map and secret contain **ObjectBucket** access information. Adding the access information to the Helm chart configuration makes it accessible to the Developer Hub container by adding the following environment variables to the container:

- **BUCKET_NAME**
- **BUCKET_HOST**
- **BUCKET_PORT**

- **BUCKET_REGION**
- **BUCKET_SUBREGION**
- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

These variables are then used in the TechDocs plugin configuration.

Prerequisites

- You have installed Red Hat Developer Hub on OpenShift Container Platform using the Helm chart.
- You have created an **ObjectBucketClaim** CR for storing files generated by TechDocs. For more information see Using OpenShift Data Foundation for file storage

Procedure

- In the **upstream.backstage** key in the Helm chart values, enter the name of the Developer Hub **ObjectBucketClaim** secret as the value for the **extraEnvVarsSecrets** field and the **extraEnvVarsCM** field. For example:

```
upstream:
  backstage:
    extraEnvVarsSecrets:
      - <rhdh_bucket_claim_name>
    extraEnvVarsCM:
      - <rhdh_bucket_claim_name>
```

2.1.2.1.1. Example TechDocs Plugin configuration for the Helm chart

The following example shows a Developer Hub Helm chart configuration for the TechDocs plugin:

```
global:
  dynamic:
    includes:
      - 'dynamic-plugins.default.yaml'
  plugins:
    - disabled: false
      package: ./dynamic-plugins/dist/backstage-plugin-techdocs-backend-dynamic
      pluginConfig:
        techdocs:
          builder: external
          generator:
            runIn: local
          publisher:
            awsS3:
              bucketName: '${BUCKET_NAME}'
              credentials:
                accessKeyId: '${AWS_ACCESS_KEY_ID}'
                secretAccessKey: '${AWS_SECRET_ACCESS_KEY}'
              endpoint: 'https://${BUCKET_HOST}'
              region: '${BUCKET_REGION}'
```

```
      s3ForcePathStyle: true
    type: awsS3
```

2.1.2.2. Making object storage accessible to containers by using the Operator

Creating a **ObjectBucketClaim** custom resource (CR) automatically generates both the Developer Hub **ObjectBucketClaim** config map and secret. The config map and secret contain **ObjectBucket** access information. Adding the access information to the Operator configuration makes it accessible to the Developer Hub container by adding the following environment variables to the container:

- **BUCKET_NAME**
- **BUCKET_HOST**
- **BUCKET_PORT**
- **BUCKET_REGION**
- **BUCKET_SUBREGION**
- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

These variables are then used in the TechDocs plugin configuration.

Prerequisites

- You have installed Red Hat Developer Hub on OpenShift Container Platform using the Operator.
- You have created an **ObjectBucketClaim** CR for storing files generated by TechDocs.

Procedure

- In your **Backstage** CR, enter the name of the Developer Hub **ObjectBucketClaim** config map as the value for the **spec.application.extraEnvs.configMaps** field and enter the Developer Hub **ObjectBucketClaim** secret name as the value for the **spec.application.extraEnvs.secrets** field. For example:

```
apiVersion: rhdh.redhat.com/v1alpha3
kind: Backstage
metadata:
 name: <name>
spec:
  application:
    extraEnvs:
      configMaps:
        - name: <rhdh_bucket_claim_name>
      secrets:
        - name: <rhdh_bucket_claim_name>
```

2.1.2.2.1. Example TechDocs Plugin configuration for the Operator

The following example shows a Red Hat Developer Hub Operator config map configuration for the TechDocs plugin:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: dynamic-plugins-rhdh
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - disabled: false
        package: ./dynamic-plugins/dist/backstage-plugin-techdocs-backend-dynamic
        pluginConfig:
          techdocs:
            builder: external
            generator:
              runIn: local
            publisher:
              awsS3:
                bucketName: '${BUCKET_NAME}'
                credentials:
                  accessKeyId: '${AWS_ACCESS_KEY_ID}'
                  secretAccessKey: '${AWS_SECRET_ACCESS_KEY}'
                endpoint: 'https://${BUCKET_HOST}'
                region: '${BUCKET_REGION}'
                s3ForcePathStyle: true
              type: awsS3
```

2.2. CONFIGURING CI/CD TO GENERATE AND PUBLISH TECHDOCS SITES

TechDocs reads the static generated documentation files from a cloud storage bucket, such as OpenShift Data Foundation. The documentation site is generated on the CI/CD workflow associated with the repository containing the documentation files. You can generate docs on CI and publish to a cloud storage using the **techdocs-cli** CLI tool.

You can use the following example to create a script for TechDocs publication:

```
# Prepare
REPOSITORY_URL='https://github.com/org/repo'
git clone $REPOSITORY_URL
cd repo

# Install @techdocs/cli, mkdocs and mkdocs plugins
npm install -g @techdocs/cli
pip install "mkdocs-techdocs-core==1.*"

# Generate
techdocs-cli generate --no-docker

# Publish
techdocs-cli publish --publisher-type awsS3 --storage-name <bucket/container> --entity
<Namespace/Kind/Name>
```

The TechDocs workflow starts the CI when a user makes changes in the repository containing the documentation files. You can configure the workflow to start only when files inside the **docs**/ directory or **mkdocs.yml** are changed.

Additional resources

- Configuring dynamic plugins

CHAPTER 3. TECHDOCS ADD-ONS

TechDocs add-ons are dynamic plugins that extend the functionality of the built-in TechDocs plugin. For example, you can use add-ons to report documentation issues, change text size, or view images in overlay in either the TechDocs Reader page or an Entity page.

The following table describes the TechDocs add-ons that are available for Red Hat Developer Hub 1.8:

Table 3.1. TechDocs Add-ons available in Red Hat Developer Hub

TechDocs Add-on	Package/Plugin	Description	Type
<ReportIssue />	**backstage-plugin-techdocs-module-addons-contrib**	Select a portion of text on a TechDocs page and open an issue against the repository that contains the documentation. The issue template is automatically populated with the selected text.	Preinstalled
<TextSize />	**backstage-plugin-techdocs-module-addons-contrib**	Customize text size on documentation pages by increasing or decreasing the font size with a slider or buttons. The default value for font size is 100% and this setting is kept in the browser's local storage whenever it is changed.	External
<LightBox />	**backstage-plugin-techdocs-module-addons-contrib**	Open images in a light-box on documentation pages, to navigate to multiple images on a single page. The image size of the light-box image is the same as the image size on the document page. Clicking the zoom icon increases the image size to fit the screen.	External

The **backstage-plugin-techdocs-module-addons-contrib** plugin package exports both preinstalled and external add-ons supported by Red Hat to the TechDocs plugin. This plugin package is preinstalled on Red Hat Developer Hub and is enabled by default. If the plugin package is disabled, all of the TechDocs add-ons exported by the package as also disabled.

3.1. INSTALLING AND CONFIGURING A TECHDOCS ADD-ON

TechDocs add-ons supported by Red Hat are exported to the TechDocs plugin by the`backstage-plugin-techdocs-module-addons-contrib` plugin package, which is preinstalled on Red Hat Developer Hub and enabled by default. The **<ReportIssue />** add-on is part of the default configuration of this plugin package and comes ready to use in the TechDocs plugin.

You can install other supported TechDocs add-ons by configuring the`backstage-plugin-techdocs-module-addons-contrib` plugin package in the Red Hat Developer Hub ConfigMap or Helm chart, depending on whether you use the Operator or Helm chart for installation. If you want to customize your TechDocs experience beyond the functions of the supported add-ons, you can install third-party add-ons on your TechDocs plugin, including add-ons that you create yourself.

3.1.1. Installing and configuring an external TechDocs add-on using the Operator

You can use a dynamic plugin to import TechDocs add-ons into your TechDocs plugin. If you use the Red Hat Developer Hub Operator to install the dynamic plugin, you can add TechDocs add-ons to the plugin package in your ConfigMap.

Preinstalled add-ons, such as **ReportIssue**, are included in the default **backstage-plugin-techdocs-module-addons-contrib** package configuration. External add-ons that are supported by Red Hat are installed by manually adding them to the **techdocsAddons** section of the configuration file.

Procedure

1. From the Developer perspective in the OpenShift Container Platform web console, click **ConfigMaps** > **Create ConfigMap**.
2. From the **Create ConfigMap** page, select the **YAML view** option in the **Configure via** field.
3. In the newly created ConfigMap, add the default **backstage-plugin-techdocs-module-addons-contrib** package configuration. For example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: dynamic-plugins-rhdh
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
```

4. In the **techdocsAddons** section of the ConfigMap, add **importName: *<external_techdocs_add-on>*** for each external TechDocs add-on that you want to add from the specified plugin package. For example:

```
kind: ConfigMap
apiVersion: v1
```

```
metadata:
  name: dynamic-plugins-rhdh
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
                  - importName: <external_techdocs_add-on>
```

where:

<external_techdocs_add-on>

Specifies the external TechDocs add-on that you want to install, for example, **TextSize** or **LightBox**.

5. Click **Create**.
6. In the web console navigation menu, click **Topology**.
7. Click on the overflow menu for the Red Hat Developer Hub instance that you want to use and select **Edit Backstage** to load the YAML view of the Red Hat Developer Hub instance.
8. In your **Backstage** CR, add the **dynamicPluginsConfigMapName: *<dynamic_plugins_configmap>*** key-value pair. For example:

```
apiVersion: rhdh.redhat.com/v1alpha3
kind: Backstage
metadata:
  name: my-rhdh
spec:
  application:
# ...
    dynamicPluginsConfigMapName: _<dynamic_plugins_configmap>_
# ...
```

where:

<dynamic_plugins_configmap>

Specifies the name of your dynamic plugins ConfigMap for your Red Hat Developer Hub instance, for example, **dynamic-plugins-rhdh**.

9. Click **Save**.
10. In the web console navigation menu, click **Topology** and wait for the Red Hat Developer Hub pod to start.

11. Click the **Open URL** icon to start using the Red Hat Developer Hub platform with the new configuration changes.

3.1.2. Installing and configuring an external TechDocs add-on using the Helm chart

You can use a dynamic plugin to import TechDocs add-ons into your TechDocs plugin. If you use the Red Hat Developer Hub Helm chart to install the dynamic plugin, you can add TechDocs add-ons to the plugin package in your Helm chart.

Preinstalled add-ons, such as **ReportIssue**, are included in the default **backstage-plugin-techdocs-module-addons-contrib** package configuration. External add-ons that are supported by Red Hat are installed by manually adding them to the **techdocsAddons** section of the configuration file.

Prerequisites

- The TechDocs plugin is installed and enabled.

Procedure

1. In your Helm chart, add the **global.dynamic** parameters required to install a dynamic plugin, as shown in Installing dynamic plugins using the Helm chart

NOTE

The default configuration includes the **dynamic-plugins.default.yaml** file, which contains all of the dynamic plugins, including TechDocs add-ons, that are preinstalled in Red Hat Developer Hub, whether they are enabled or disabled by default.

2. In your Helm chart, add the default **backstage-plugin-techdocs-module-addons-contrib** package configuration. For example:

```
global:
  dynamic:
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
```

3. In the **techdocsAddons** section of the Helm chart, add **importName: *<external_techdocs_add-on>*** for each external TechDocs add-on that you want to add from the specified plugin package. For example:

```
global:
  dynamic:
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
```

```
dynamicPlugins:
  frontend:
    backstage.plugin-techdocs-module-addons-contrib:
      techdocsAddons:
        - importName: ReportIssue
        - importName: <external_techdocs_add-on>
```

where:

<external_techdocs_add-on>

Specifies the external TechDocs add-on that you want to install, for example, **TextSize** or **LightBox**.

3.1.3. Installing and configuring a third-party TechDocs add-on

You can install compatible third-party TechDocs add-on on your Red Hat Developer Hub instance as a front-end dynamic plugin.

Prerequisites

- The third-party TechDocs add-on has a valid **package.json** file in its root directory, containing all required metadata and dependencies.
- The third-party plugin is packaged as a dynamic plugin in an OCI image. For alternative package types, see Installing third-party plugins in Red Hat Developer Hub .
- You have installed the **yarn** package manager.
- The third-party plugin is packaged as a dynamic plugin in an OCI image.* You have installed and configured Node.js and NPM.

Procedure

1. Install the third-party plugin that you want to use to import your third-party add-on by entering the following command:

```
$ yarn install
```

2. Obtain the source code for the third-party TechDocs add-on that you want to use.
3. Export the TechDocs add-on as a dynamic plugin using the following command:

```
$ npx @red-hat-developer-hub/cli@latest plugin export
```

NOTE

The **@latest** tag pulls the latest version of the @red-hat-developer-hub/cli package, which is compatible with the most recent features and fixes. Use a version that is compatible with your Red Hat Developer Hub version.

4. To package the third-party TechDocs add-on as a dynamic plugin, navigate to the root directory where the plugin is stored (not the dist-dynamic directory) and run the **npx** command with the **--tag** option to specify the image name and tag. For example:

```
$ npx @red-hat-developer-hub/cli@latest plugin package --tag
quay.io/<user_name>/<techdocs_add-on_image>:latest
```

NOTE

The output of the package-dynamic-plugins command provides the file path to the plugin for use in the **dynamic-plugin-config.yaml** file.

5. To publish the third-party TechDocs add-on to a Quay repository, push the image to a registry using one of the following commands, depending on your virtualization tool:

   - To use **podman**, enter the following command:

     ```
     $ podman push quay.io/<user_name>/<techdocs_add-on_image>:latest
     ```

   - To use **docker**, enter the following command:

     ```
     $ docker push quay.io/<user_name>/<techdocs_add-on_image>:latest
     ```

6. Open your **dynamic-plugins.yaml** file to view or modify the configuration for the third-party TechDocs add-on. For example:

```
plugins:
  - package: oci://quay.io/<user_name>/<techdocs_add-on_image>:latest!<techdocs_add-on_package>
    disabled: false
    pluginConfig:
      dynamicPlugins:
        frontend:
          <techdocs_add-on_package>
            techdocsAddons:
              - importName: <third-party_add-on_name>
                config:
                  props:
                    <techdocs_add-on_property_key>: <techdocs_add-on_property_value>
```

where

<user_name>

Specifies your Quay user name or organization name.

<techdocs_add-on_image>

Specifies the name of the image for the third-party add-on that you want to use, for example, **mermaid**.

<techdocs_add-on_package>

Specifies the , for example, **backstage-plugin-techdocs-addon-mermaid**.

<third-party_add-on_name>

Specifies the name of the third-party add-on that you want to use, for example, **Mermaid**.

<techdocs_add-on_property_key>

Specifies the name of the custom property that can be passed to the third-party add-on, for example, **themeVariables**. Properties are specific to each add-on. You can list multiple properties for an add-on.

<techdocs_add-on_property_value>

Specifies the value of a property key for the third-party add-on, for example, **lineColor: #000000**.

Additional resources

- Configuring dynamic plugins
- Installing and viewing plugins in Red Hat Developer Hub

3.2. REMOVING A TECHDOCS ADD-ON

Administrators can remove installed TechDocs add-ons from your Red Hat Developer Hub instance by using either the Operator or Helm chart, depending on the method used to install the add-on. If you used the Operator to install the add-on, remove it from the ConfigMap. If you used the Helm chart to install the add-on, remove it from the Helm chart.

If you want to disable a plugin instead of removing it from your Red Hat Developer Hub instance, you can disable the plugin that you are using to import the TechDocs add-on. Since the **disabled** status is controlled at the plugin level, disabling the plugin disables all of the TechDocs add-ons in the specified plugin package.

3.2.1. Removing an external TechDocs add-on from your ConfigMap

If you no longer want to use the functionality of a TechDocs add-on that is imported from a particular plugin that you installed on your Red Hat Developer Hub instance with the Operator, you can temporarily disable it or permanently remove it from your ConfigMap. The **disabled** status is controlled at the plugin level, therefore, disabling the plugin disables all of the TechDocs add-ons in the disabled plugin package.

Procedure

1. From the Developer perspective in the OpenShift Container Platform web console, click **ConfigMaps**.
2. From the **ConfigMaps** page, click the ConfigMap that contains the TechDocs add-on that you want to remove.
3. Select the **YAML view** option in the **Configure via** field.
4. In the **plugins** section of the ConfigMap, do one of the following actions based on whether you want to disable or remove a TechDocs add-on:
   - To temporarily disable all of the TechDocs add-ons in a particular plugin package, change the value of the **disabled** field to **true**. For example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: dynamic-plugins-rhdh
data:
```

```
dynamic-plugins.yaml: |
  includes:
    - dynamic-plugins.default.yaml
  plugins:
    - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
      disabled: true
      pluginConfig:
        dynamicPlugins:
          frontend:
            backstage.plugin-techdocs-module-addons-contrib:
              techdocsAddons:
                - importName: ReportIssue
                - importName: <external_techdocs_add-on>
```

where:

<external_techdocs_add-on>

Specifies the external TechDocs add-on that you want to remove, for example, **TextSize**.

- To remove one or more TechDocs add-ons from your Red Hat Developer Hub instance, delete **importName: *<external_techdocs_add-on>*** for each external TechDocs add-on that you want to remove from the **techdocsAddons** section of your ConfigMap. For example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: dynamic-plugins-rhdh
data:
  dynamic-plugins.yaml: |
    includes:
      - dynamic-plugins.default.yaml
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
                  - importName: <external_techdocs_add-on>
```

where:

<external_techdocs_add-on>

Specifies the external TechDocs add-on that you want to remove, for example, **TextSize**.

5. Click **Save**.
6. In the web console navigation menu, click **Topology** and wait for the Red Hat Developer Hub pod to start.
7. Click the **Open URL** icon to start using the Red Hat Developer Hub platform with the new configuration changes.

3.2.2. Removing an external TechDocs add-on from your Helm chart

If you no longer want to use the functionality of a TechDocs add-on that is imported from a particular plugin that you installed on your Red Hat Developer Hub instance with the Helm chart, you can temporarily disable it or permanently remove it from your Helm chart. The **disabled** status is controlled at the plugin level, therefore, disabling the plugin disables all of the TechDocs add-ons in the disabled plugin package.

Procedure

- In the **plugins** section of the Helm chart, do one of the following actions based on whether you want to disable or remove a TechDocs add-on:
  - To temporarily disable all of the TechDocs add-ons in a particular plugin package, change the value of the **disabled** field to **true**. For example:

```
global:
  dynamic:
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: true
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
                  - importName: <external_techdocs_add-on>
```

    where:

    <external_techdocs_add-on>

    Specifies the external TechDocs add-on that you want to remove, for example, **TextSize**.

  - To remove one or more TechDocs add-ons from your Red Hat Developer Hub instance, delete **importName: *<external_techdocs_add-on>*** for each external TechDocs add-on that you want to remove from the **techdocsAddons** section of your Helm chart. For example:

```
global:
  dynamic:
    plugins:
      - package: ./dynamic-plugins/dist/backstage-plugin-techdocs-module-addons-contrib
        disabled: false
        pluginConfig:
          dynamicPlugins:
            frontend:
              backstage.plugin-techdocs-module-addons-contrib:
                techdocsAddons:
                  - importName: ReportIssue
                  - importName: <external_techdocs_add-on>
```

    where:

    <external_techdocs_add-on>

Specifies the external TechDocs add-on that you want to remove, for example, **TextSize**.

3.3. USING TECHDOCS ADD-ONS

After an administrator installs a TechDocs add-on in your Red Hat Developer Hub instance, you can use it to extend the functionality of the TechDocs plugin and enhance your documentation experience.

3.3.1. Using the ReportIssue TechDocs add-on

If you find an error in your organization's TechDocs documentation, you can use the **ReportIssue** add-on to open a new GitHub or GitLab issue directly from the documentation. **ReportIssue** automatically imports the text that you highlight in the document into a new issue template in your repository.

Prerequisites

- The **ReportIssue** add-on is installed and enabled in your TechDocs plugin.
- You have permissions to create issues in the repository where documentation issues are reported.

Procedure

1. In your TechDocs documentation, highlight the text that you want to report an issue for.
2. Click the text in the **ReportIssue** box, for example, **Open new GitHub issue**
3. From the new issue page in your repository, use the template to create the issue that you want to report.

NOTE

The default issue title is **Documentation feedback: *<highlighted_text>***, where *<highlighted_text>* is the text that you highlighted in your TechDocs documentation.

In the issue description, *<highlighted_text>* is the default value for the **The highlighted text** field.

Verification

- The issue that you created is listed on the issues page in your repository.

3.3.2. Using the TextSize TechDocs add-on

You can use the **TextSize** add-on to change the size of the text on either the TechDocs Reader page or an Entity page.

Prerequisites

- The **TextSize** add-on is installed and enabled in your TechDocs plugin.

Procedure

1. In your TechDocs header, click the **Settings** icon.
2. Use the sliding scale to adjust the size of your documentation text.

NOTE

- The default text size is 100%
- The minimize text size is 90%
- The maximum text size is 150%

3.3.3. Using the LightBox TechDocs add-on

If your TechDocs documentation contains an image, you can use the **LightBox** add-on to view an enlarged version of the image in a lightbox, or overlay window. You can also zoom to change the size the lightbox image. If a single documentation page contains multiple images, you can navigate between images in the lightbox.

Prerequisites

- The **LightBox** add-on is installed and enabled in your TechDocs plugin.

Procedure

1. In your TechDocs documentation, click on the image that you want to view in a lightbox.
2. In the lightbox, you can do any of the following actions:
   - Click the image or scroll to zoom in or zoom out.
   - Click the arrow to navigate between images.

CHAPTER 4. CREATING A TECHDOCS ADD-ON

If your organization has documentation needs that are not met by the functions of existing TechDocs add-ons, developers can create a new add-on for your TechDocs plugin.

A TechDocs add-on is a React component that is imported from a front-end plugin. If you do not have an existing plugin that you can use to export your TechDocs add-on, you can create a new plugin by using **backstage-cli** to generate a default front-end plugin structure that you can customize.

The folder structure of a new plugin that can be used to import TechDocs add-ons into the TechDocs plugin looks similar to the following example:

```
<new_plugin_for_techdocs_add-on>/
  dev/
     index.ts
  src/
     components/
       <new_techdocs_add-on_component>/
            <new_techdocs_add-on_component>.test.tsx
            <new_techdocs_add-on_component>.tsx
             index.ts
       <new_techdocs_add-on_fetch-component>/
            <new_techdocs_add-on_fetch-component>.test.tsx
            <new_techdocs_add-on_fetch-component>.tsx
             index.ts
     index.ts
     plugin.test.ts
     plugin.ts
     routes.ts
     setupTests.ts
  .eslintrc.js
  package.json
  README.md
```

Prerequisites

- The **yarn** package manager is installed.
- Docker v3.2.0 or later or Podman v3.2.0 or later is installed and running.

Procedure

1. In the terminal, navigate to the root folder of the repository where you want to create your new plugin.
2. To create a new front-end plugin, run the following command:

   ```
   $ yarn new
   ```

   Example output:

   ```
   ? What do you want to create? plugin - A new frontend plugin
   ? Enter the ID of the plugin [required]
   ```

3. In the terminal prompt, type a name for the new plugin. For example:

```
? Enter the ID of the plugin [required] <new_plugin_for_techdocs_add-on>
```

Example output

```
Successfully created plugin
```

Upon completion of this action, a sub-directory with the same name that you gave to your plugin is automatically generated inside the **plugins** directory. The directory contains all of the files that you need to configure to create your new plugin.

4. In the terminal, navigate to your new plugin directory. For example:

```
$ cd plugins/<new_techdocs_add-on_directory>
```

5. Add the `@backstage/plugin-techdocs-react` package to get frontend utilities for TechDocs add-ons. For example:

```
$ yarn add @backstage/plugin-techdocs-react
```

6. In the directory containing the components of your custom TechDocs add-on, delete any default files or file components that are not required for your add-on, such as the **routes.ts** file or components of the **index.tsx** and **plugins.ts** files.

7. In the **plugins.ts** file, add the following code:

```
$ import { createPlugin } from '@backstage/core-plugin-api';
$ import { createTechDocsAddonExtension } from '@backstage/plugin-techdocs-react';

$ export const <new_plugin_for_techdocs_add-on> = createPlugin({
  id: '<new_techdocs_add-on>',
 });

 /*
  *
  * @public
  */
$ export const <new_techdocs_add-on> = <new_plugin_for_techdocs_add-on>.provide(
 createTechDocsAddonExtension<_<new_techdocs_addon_props>_>({
    name: '<new_techdocs_add-on>',
    location: TechDocsAddonLocations.Content,
    component: <new_techdocs_add-on_component>,
  }),
);
```

where

<new_plugin_for_techdocs_add-on>

Specifies the new plugin that you use to import the TechDocs add-on to your Red Hat Developer Hub instance.

<new_techdocs_add-on>

Specifies the custom TechDocs add-on that you want to create.

<new_techdocs_addon_props> (Optional)

Specifies the **props** for your new TechDocs add-on, as specified in your ***<new_techdocs_add-on>*.tsx** file, if applicable.

<new_techdocs_add-on_component>

Specifies the React component for the custom TechDocs add-on that you want to create. You will create this component in a **.tsx** file in a later step.

8. In the **index.ts** file, export the custom TechDocs add-on that you want to create by adding the following code:

```
$ export { <new_plugin_for_techdocs_add-on>, <new_techdocs_add-on> } from './plugin';
```

9. Create a new ***<new_techdocs_add-on>*.tsx** file and add the code for your new TechDocs add-on component.

10. Create a new **index.tsx** file and use it to export your new TechDocs add-on component by adding the following code:

```
$ export { <new_techdocs_add-on>, type <new_techdocs_addon_props>} from
'./<new_techdocs_add-on_directory>'
```

where

<new_techdocs_addon_props> (Optional)

Specifies the **props** for your new TechDocs add-on, as specified in your ***<new_techdocs_add-on>*.tsx** file, if applicable.

11. In the **plugins.ts** file, import your new TechDocs add-on component.

12. Install and configure your new TechDocs add-on by following the steps in Installing and configuring a TechDocs add-on

Verification

1. Restart the RHDH application and verify that the plugin is successfully activated and configured.

2. Verify the application logs for confirmation and ensure the plugin is functioning as expected.