OpenShift Container Platform 4.10

Web console

Getting started with the web console in OpenShift Container Platform
Getting started with the web console in OpenShift Container Platform
Abstract

This document provides instructions for accessing and customizing the OpenShift Container Platform web console.
# Table of Contents

CHAPTER 1. ACCESSING THE WEB CONSOLE ......................................................... 4
   1.1. PREREQUISITES ............................................................................. 4
   1.2. UNDERSTANDING AND ACCESSING THE WEB CONSOLE ............. 4

CHAPTER 2. USING THE OPENSİFT CONTAINER PLATFORM DASHBOARD TO GET CLUSTER INFORMATION .......................................................... 5
   2.1. ABOUT THE OPENSİFT CONTAINER PLATFORM DASHBOARDS PAGE ......................................................................................... 5

CHAPTER 3. ADDING USER PREFERENCES ......................................................... 7
   3.1. SETTING USER PREFERENCES ......................................................... 7

CHAPTER 4. CONFIGURING THE WEB CONSOLE IN OPENSİFT CONTAINER PLATFORM .......................................................... 8
   4.1. PREREQUISITES ............................................................................. 8
   4.2. CONFIGURING THE WEB CONSOLE ............................................... 8

CHAPTER 5. CUSTOMIZING THE WEB CONSOLE IN OPENSİFT CONTAINER PLATFORM .......................................................... 9
   5.1. ADDING A CUSTOM LOGO AND PRODUCT NAME ......................... 9
   5.2. CREATING CUSTOM LINKS IN THE WEB CONSOLE ................. 10
   5.3. CUSTOMIZING CONSOLE ROUTES ............................................... 11
      5.3.1. Customizing the console route ................................................. 11
      5.3.2. Customizing the download route ............................................ 12
   5.4. CUSTOMIZING THE LOGIN PAGE ................................................ 13
   5.5. DEFINING A TEMPLATE FOR AN EXTERNAL LOG LINK ............. 14
   5.6. CREATING CUSTOM NOTIFICATION BANNERS ......................... 15
   5.7. CUSTOMIZING CLI DOWNLOADS ................................................ 15
   5.8. ADDING YAML EXAMPLES TO KUBERNETES RESOURCES ......... 16

CHAPTER 6. ADDING A DYNAMIC PLUG-IN TO THE OPENSİFT CONTAINER PLATFORM WEB CONSOLE .......................................................... 18
   6.1. ABOUT DYNAMIC PLUG-INS ......................................................... 18
      6.1.1. Key features ......................................................................... 18
      6.1.2. PatternFly 4 guidelines .......................................................... 18
      6.1.3. General guidelines ............................................................... 18
   6.2. GETTING STARTED WITH DYNAMIC PLUG-INS ....................... 19
   6.3. RUNNING YOUR DYNAMIC PLUG-IN .......................................... 19
      6.3.1. Adding a tab to the pods page ............................................... 20
   6.4. ADDING A NEW EXTENSION TO YOUR PLUG-IN .................... 21
   6.5. BUILD AN IMAGE WITH DOCKER .............................................. 21
   6.6. DEPLOY YOUR PLUG-IN ON A CLUSTER .................................... 21
   6.7. ADDITIONAL RESOURCES ......................................................... 22

CHAPTER 7. ABOUT THE DEVELOPER PERSPECTIVE IN THE WEB CONSOLE .......................................................... 23
   7.1. PREREQUISITES ............................................................................. 23
   7.2. ACCESSING THE DEVELOPER PERSPECTIVE ............................ 23

CHAPTER 8. ABOUT THE WEB TERMINAL IN THE WEB CONSOLE .......................................................... 25
   8.1. INSTALLING THE WEB TERMINAL .............................................. 25
   8.2. USING THE WEB TERMINAL ....................................................... 26
   8.3. UNINSTALLING THE WEB TERMINAL ......................................... 26
      8.3.1. Removing the Web Terminal Operator and the custom resources that support it .......................................................... 27
      8.3.2. Deleting the DevWorkspace Operator dependency ................. 27

CHAPTER 9. DISABLING THE WEB CONSOLE IN OPENSİFT CONTAINER PLATFORM .......................................................... 31
   9.1. PREREQUISITES ............................................................................. 31
CHAPTER 1. ACCESSING THE WEB CONSOLE

The OpenShift Container Platform web console is a user interface accessible from a web browser. Developers can use the web console to visualize, browse, and manage the contents of projects.

1.1. PREREQUISITES

- JavaScript must be enabled to use the web console. For the best experience, use a web browser that supports WebSockets.
- Review the OpenShift Container Platform 4.x Tested Integrations page before you create the supporting infrastructure for your cluster.

1.2. UNDERSTANDING AND ACCESSING THE WEB CONSOLE

The web console runs as a pod on the master. The static assets required to run the web console are served by the pod. After OpenShift Container Platform is successfully installed using openshift-install create cluster, find the URL for the web console and login credentials for your installed cluster in the CLI output of the installation program. For example:

Example output

```
INFO Install complete!
INFO Run 'export KUBECONFIG=<your working directory>/auth/kubeconfig' to manage the cluster with 'oc', the OpenShift CLI.
INFO The cluster is ready when 'oc login -u kubeadmin -p <provided>' succeeds (wait a few minutes).
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.demo1.openshift4-beta-abcorp.com
INFO Login to the console with user: kubeadmin, password: <provided>
```

Use those details to log in and access the web console.

For existing clusters that you did not install, you can use `oc whoami --show-console` to see the web console URL.
Access the OpenShift Container Platform dashboard, which captures high-level information about the cluster, by navigating to Home → Dashboards → Overview from the OpenShift Container Platform web console.

The OpenShift Container Platform dashboard provides various cluster information, captured in individual dashboard cards.

2.1. ABOUT THE OPENSHIFT CONTAINER PLATFORM DASHBOARDS PAGE

The OpenShift Container Platform dashboard consists of the following cards:

- **Details** provides a brief overview of informational cluster details. Status include **ok**, **error**, **warning**, **in progress**, and **unknown**. Resources can add custom status names.
  - Cluster ID
  - Provider
  - Version

- **Cluster Inventory** details number of resources and associated statuses. It is helpful when intervention is required to resolve problems, including information about:
  - Number of nodes
  - Number of pods
  - Persistent storage volume claims
  - Bare metal hosts in the cluster, listed according to their state (only available in metal3 environment).

- **Cluster Capacity** charts help administrators understand when additional resources are required in the cluster. The charts contain an inner ring that displays current consumption, while an outer ring displays thresholds configured for the resource, including information about:
  - CPU time
  - Memory allocation
  - Storage consumed
  - Network resources consumed

- **Cluster Utilization** shows the capacity of various resources over a specified period of time, to help administrators understand the scale and frequency of high resource consumption.

- **Events** lists messages related to recent activity in the cluster, such as pod creation or virtual machine migration to another host.
- **Top Consumers** helps administrators understand how cluster resources are consumed. Click on a resource to jump to a detailed page listing pods and nodes that consume the largest amount of the specified cluster resource (CPU, memory, or storage).
CHAPTER 3. ADDING USER PREFERENCES

You can change the default preferences for your profile to meet your requirements. You can set your default project, topology view (graph/list), editing medium (form or YAML), and language preferences.

The changes made to the user preferences are automatically saved.

3.1. SETTING USER PREFERENCES

You can set the default user preferences for your cluster.

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials.

2. Use the masthead to access the user preferences under the user profile.

3. In the General section:
   a. In the Perspective field, you can set the default perspective you want to be logged in to. You can select the Administrator or the Developer perspective as required. If a perspective is not selected, you are logged into the perspective you last visited.
   b. In the Project field, select a project you want to work in. The console will default to the project every time you log in.
   c. In the Topology field, you can set the topology view to default to the graph or list view. If not selected, the console defaults to the last view you used.
   d. In the Create/Edit resource method field, you can set a preference for creating or editing a resource. If both the form and YAML options are available, the console defaults to your selection.

4. In the Language section, select Default browser language to use the default browser language settings. Otherwise, select the language that you want to use for the console.
CHAPTER 4. CONFIGURING THE WEB CONSOLE IN OPENSHIFT CONTAINER PLATFORM

You can modify the OpenShift Container Platform web console to set a logout redirect URL or disable the console.

4.1. PREREQUISITES

- Deploy an OpenShift Container Platform cluster.

4.2. CONFIGURING THE WEB CONSOLE

You can configure the web console settings by editing the `console.config.openshift.io` resource.

- Edit the `console.config.openshift.io` resource:

  ```bash
  $ oc edit console.config.openshift.io cluster
  
  apiVersion: config.openshift.io/v1
  kind: Console
  metadata:
    name: cluster
  spec:
    authentication:
      logoutRedirect: ""  
  status:
    consoleURL: ""

  1 Specify the URL of the page to load when a user logs out of the web console. If you do not specify a value, the user returns to the login page for the web console. Specifying a `logoutRedirect` URL allows your users to perform single logout (SLO) through the identity provider to destroy their single sign-on session.

  2 The web console URL. To update this to a custom value, see Customizing the web console URL.
CHAPTER 5. CUSTOMIZING THE WEB CONSOLE IN OPENSHIFT CONTAINER PLATFORM

You can customize the OpenShift Container Platform web console to set a custom logo, product name, links, notifications, and command line downloads. This is especially helpful if you need to tailor the web console to meet specific corporate or government requirements.

5.1. ADDING A CUSTOM LOGO AND PRODUCT NAME

You can create custom branding by adding a custom logo or custom product name. You can set both or one without the other, as these settings are independent of each other.

Prerequisites

- You must have administrator privileges.
- Create a file of the logo that you want to use. The logo can be a file in any common image format, including GIF, JPG, PNG, or SVG, and is constrained to a max-height of 60px.

Procedure

1. Import your logo file into a config map in the openshift-config namespace:

```
$ oc create configmap console-custom-logo --from-file /path/to/console-custom-logo.png -n openshift-config
```

TIP

You can alternatively apply the following YAML to create the config map:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: console-custom-logo
  namespace: openshift-config
data:
  console-custom-logo.png: <base64-encoded_logo> ...
```

Provide a valid base64-encoded logo.

2. Edit the web console’s Operator configuration to include customLogoFile and customProductName:

```
$ oc edit consoles.operator.openshift.io cluster
```

```yaml
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
customization:
customLogoFile:
```

Once the Operator configuration is updated, it will sync the custom logo config map into the console namespace, mount it to the console pod, and redeploy.

3. Check for success. If there are any issues, the console cluster Operator will report a **Degraded** status, and the console Operator configuration will also report a **CustomLogoDegraded** status, but with reasons like **KeyOrFilenameInvalid** or **NoImageProvided**.

To check the **clusteroperator**, run:

```
$ oc get clusteroperator console -o yaml
```

To check the console Operator configuration, run:

```
$ oc get consoles.operator.openshift.io -o yaml
```

### 5.2. CREATING CUSTOM LINKS IN THE WEB CONSOLE

**Prerequisites**

- You must have administrator privileges.

**Procedure**

1. From **Administration → Custom Resource Definitions** click on **ConsoleLink**.

2. Select **Instances** tab

3. Click **Create Console Link** and edit the file:

```
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: example
spec:
  href: 'https://www.example.com'
  location: HelpMenu
  text: Link
```

Valid location settings are **HelpMenu, UserMenu, ApplicationMenu**, and **NamespaceDashboard**.

To make the custom link appear in all namespaces, follow this example:

```
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: namespaced-dashboard-link-for-all-namespaces
spec:
```

```
To make the custom link appear in only some namespaces, follow this example:

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: namespaced-dashboard-for-some-namespaces
spec:
  href: 'https://www.example.com'
  location: NamespaceDashboard
  text: Custom Link Text
  namespaceDashboard:
    namespaces:
      # for these specific namespaces
      - my-namespace
      - your-namespace
      - other-namespace
```

To make the custom link appear in the application menu, follow this example:

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: application-menu-link-1
spec:
  href: 'https://www.example.com'
  location: ApplicationMenu
  text: Link
  applicationMenu:
    section: My New Section
    # image that is 24x24 in size
    imageURL: https://via.placeholder.com/24
```

4. Click **Save** to apply your changes.

### 5.3. CUSTOMIZING CONSOLE ROUTES

For console and downloads routes, custom routes functionality uses the `ingress` config route configuration API. If the console custom route is set up in both the `ingress` config and `console-operator` config, then the new `ingress` config custom route configuration takes precedent. The route configuration with the `console-operator` config is deprecated.

#### 5.3.1. Customizing the console route

You can customize the console route by setting the custom hostname and TLS certificate in the `spec.componentRoutes` field of the cluster `Ingress` configuration.

**Prerequisites**
- You have logged in to the cluster as a user with administrative privileges.

- You have created a secret in the `openshift-config` namespace containing the TLS certificate and key. This is required if the domain for the custom hostname suffix does not match the cluster domain suffix. The secret is optional if the suffix matches.

**TIP**

You can create a TLS secret by using the `oc create secret tls` command.

**Procedure**

1. Edit the cluster `Ingress` configuration:

```
$ oc edit ingress.config.openshift.io cluster
```

2. Set the custom hostname and optionally the serving certificate and key:

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  componentRoutes:
    - name: console
      namespace: openshift-console
      hostname: <custom_hostname>  
      servingCertKeyPairSecret:
        name: <secret_name>
```

1. The custom hostname.

2. Reference to a secret in the `openshift-config` namespace that contains a TLS certificate (`tls.crt`) and key (`tls.key`). This is required if the domain for the custom hostname suffix does not match the cluster domain suffix. The secret is optional if the suffix matches.

3. Save the file to apply the changes.

**5.3.2. Customizing the download route**

You can customize the download route by setting the custom hostname and TLS certificate in the `spec.componentRoutes` field of the cluster `Ingress` configuration.

**Prerequisites**

- You have logged in to the cluster as a user with administrative privileges.

- You have created a secret in the `openshift-config` namespace containing the TLS certificate and key. This is required if the domain for the custom hostname suffix does not match the cluster domain suffix. The secret is optional if the suffix matches.
TIP

You can create a TLS secret by using the `oc create secret tls` command.

Procedure

1. Edit the cluster **Ingress** configuration:

   ```shell
   $ oc edit ingress.config.openshift.io cluster
   ```

2. Set the custom hostname and optionally the serving certificate and key:

   ```yaml
   apiVersion: config.openshift.io/v1
   kind: Ingress
   metadata:
     name: cluster
   spec:
     componentRoutes:
       - name: downloads
         namespace: openshift-console
         hostname: <custom_hostname>  
         servingCertKeyPairSecret:
           name: <secret_name>
   ```

   1. The custom hostname.
   2. Reference to a secret in the `openshift-config` namespace that contains a TLS certificate (`tls.crt`) and key (`tls.key`). This is required if the domain for the custom hostname suffix does not match the cluster domain suffix. The secret is optional if the suffix matches.

3. Save the file to apply the changes.

5.4. CUSTOMIZING THE LOGIN PAGE

Create Terms of Service information with custom login pages. Custom login pages can also be helpful if you use a third-party login provider, such as GitHub or Google, to show users a branded page that they trust and expect before being redirected to the authentication provider. You can also render custom error pages during the authentication process.

NOTE

Customizing the error template is limited to identity providers (IDPs) that use redirects, such as request header and OIDC-based IDPs. It does not have an effect on IDPs that use direct password authentication, such as LDAP and HTPasswd.

Prerequisites

- You must have administrator privileges.

Procedure

1. Run the following commands to create templates you can modify:
2. Create the secrets:

```bash
$ oc adm create-login-template > login.html
$ oc adm create-provider-selection-template > providers.html
$ oc adm create-error-template > errors.html
```

3. Run:

```bash
$ oc edit oauths cluster
```

4. Update the specification:

```json
spec:
  templates:
    error:
      name: error-template
    login:
      name: login-template
    providerSelection:
      name: providers-template
```

Run `oc explain oauths.spec.templates` to understand the options.

5.5. DEFINING A TEMPLATE FOR AN EXTERNAL LOG LINK

If you are connected to a service that helps you browse your logs, but you need to generate URLs in a particular way, then you can define a template for your link.

**Prerequisites**
- You must have administrator privileges.

**Procedure**

1. From Administration → Custom Resource Definitions click on ConsoleExternalLogLink.

2. Select Instances tab

3. Click Create Console External Log Link and edit the file:

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleExternalLogLink
metadata:
```
### 5.6. CREATING CUSTOM NOTIFICATION BANNERS

**Prerequisites**

- You must have administrator privileges.

**Procedure**

1. From Administration → Custom Resource Definitions, click on ConsoleNotification.
2. Select Instances tab
3. Click Create Console Notification and edit the file:
   ```yaml
   apiVersion: console.openshift.io/v1
   kind: ConsoleNotification
   metadata:
     name: example
   spec:
     text: This is an example notification message with an optional link.
     location: BannerTop
     link:
       href: 'https://www.example.com'
       text: Optional link text
       color: '#fff'
       backgroundColor: '#0088ce'
   ``

   Valid location settings are BannerTop, BannerBottom, and BannerTopBottom.
4. Click Create to apply your changes.

### 5.7. CUSTOMIZING CLI DOWNLOADS

You can configure links for downloading the CLI with custom link text and URLs, which can point directly to file packages or to an external page that provides the packages.

**Prerequisites**

- You must have administrator privileges.

**Procedure**

1. Navigate to Administration → Custom Resource Definitions
2. Select ConsoleCLIDownload from the list of Custom Resource Definitions (CRDs).
3. Click the **YAML** tab, and then make your edits:

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleCLIDownload
metadata:
  name: example-cli-download-links-for-foo
spec:
  description: |
    This is an example of download links for foo
displayName: example-foo
links:
    text: foo for linux
    text: foo for mac
    text: foo for windows
```

4. Click the **Save** button.

### 5.8. ADDING YAML EXAMPLES TO KUBERNETES RESOURCES

You can dynamically add YAML examples to any Kubernetes resources at any time.

**Prerequisites**

- You must have cluster administrator privileges.

**Procedure**

1. From **Administration → Custom Resource Definitions**, click on **ConsoleYAMLSample**.

2. Click **YAML** and edit the file:

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleYAMLSample
metadata:
  name: example
spec:
targetResource:
  apiVersion: batch/v1
  kind: Job
title: Example Job
description: An example Job YAML sample
yaml: |
  apiVersion: batch/v1
  kind: Job
metadata:
  name: countdown
spec:
template:
  metadata:
    name: countdown
spec:
  containers:
```
Use **spec.snippet** to indicate that the YAML sample is not the full YAML resource definition, but a fragment that can be inserted into the existing YAML document at the user’s cursor.

3. Click **Save**.
CHAPTER 6. ADDING A DYNAMIC PLUG-IN TO THE OPENSHIFT CONTAINER PLATFORM WEB CONSOLE

You can create and deploy a dynamic plug-in on your cluster that is loaded at run-time.

IMPORTANT

Creating a dynamic plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in 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 about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

6.1. ABOUT DYNAMIC PLUG-INS

A dynamic plug-in allows you to add custom pages and other extensions to your interface at runtime. The ConsolePlugin custom resource registers plug-ins with the console, and a cluster administrator enables plug-ins in the console-operator configuration.

6.1.1. Key features

A dynamic plug-in allows you to make the following customizations to the OpenShift Container Platform experience:

- Add custom pages.
- Add perspectives and update navigation items.
- Add tabs and actions to resource pages.

6.1.2. PatternFly 4 guidelines

When creating your plug-in, follow these guidelines for using PatternFly:

- Use PatternFly4 components and PatternFly CSS variables. Core PatternFly components are available through the SDK. Using PatternFly components and variables will help your plug-in look consistent in future console versions.
- Make your plug-in accessible by following PatternFly’s accessibility fundamentals.
- Do not use other CSS libraries such as Bootstrap or Tailwind. They can conflict with PatternFly and will not match the console look and feel.

6.1.3. General guidelines

When creating your plug-in, follow these general guidelines:

- Prefix your CSS class names with your plug-in name to avoid collisions. For example, my-plugin__heading and my-plugin__icon.
- Maintain a consistent look, feel, and behavior with other console pages.
Use react-i18next for localization.

Do not use console CSS classes in your markup or override console CSS classes. These are not APIs and are subject to change. Using them might break your plug-in. Avoid selectors like element selectors that could affect markup outside of your plug-in's components.

6.2. GETTING STARTED WITH DYNAMIC PLUG-INS

To get started using the dynamic plug-in, you must set up your environment to write a new OpenShift Console dynamic plug-in.

Prerequisites

- Ensure you have Node.js installed.
- Ensure you have yarn installed.

Procedure

1. Visit this repository containing a template for creating plug-ins.
2. Select Use this Template from the <> Code tab to create a GitHub repository.
3. Re-name the template with the name of your plug-in.
4. From your copied repository, clone it your local machine so you can edit the code.
5. Edit the plug-in metadata in the consolePlugin declaration of package.json.

```
"consolePlugin": {
  "name": "my-plugin",  // 1
  "version": "0.0.1",  // 2
  "displayName": "My Plugin",  // 3
  "description": "Enjoy this shiny, new console plugin!",  // 4
  "exposedModules": {
    "ExamplePage": "./components/ExamplePage"
  },
  "dependencies": {
    "@console/pluginAPI": "**"
  }
}
```

- Update the name of your plug-in.
- Update the version.
- Update the display name for your plug-in.
- Update the description with a synopsis about your plug-in.

6.3. RUNNING YOUR DYNAMIC PLUG-IN
You can run the plug-in using a local development environment. The OpenShift console runs in a container connected to the cluster you have logged into.

Prerequisites

- You must have the OpenShift CLI (oc) installed.
- You must have an OpenShift cluster running.
- You must have Docker or at least v3.2.0 of Podman installed.

Procedure

- Open two terminal windows in the local directory of your cloned repository.
  - Run the following commands in the first terminal:
    ```
    $ yarn install
    $ yarn run start
    ```
  - Run the following commands in the second terminal window:
    ```
    $ oc login
    $ yarn run start-console
    ```

Verification

- Visit local host to view the running plug-in.

6.3.1. Adding a tab to the pods page

The following procedure adds a tab to the Pod Details page as an example extension to your plug-in.

Procedure

1. Add the following to the console-extensions.json file:

   ```
   {
   "type": "console.page/resource/tab",
   "properties": {
      "name": "Example Tab",
      "href": "example",
      "model": {
         "group": "core",
         "version": "v1",
         "kind": "Pod"
      },
      "component": {
         "$codeRef": "ExampleTab"
      }
   }
   }
   ```

2. Edit the package.json file to include the following changes:
3. Write a message to display on a new custom tab on the Pods page by creating a new file `src/components/ExampleTab.tsx` and adding the following script:

```jsx
import * as React from 'react';

export default function ExampleTab() {
  return (<p>This is a custom tab added to a resource using a dynamic plug-in.</p>);
}
```

**Verification**

- Visit a Pod page to view the added tab.

### 6.4. ADDING A NEW EXTENSION TO YOUR PLUG-IN

You can add extensions to your plug-in that are loaded at runtime.

### 6.5. BUILD AN IMAGE WITH DOCKER

To deploy your plug-in on a cluster, you need to build an image and push it to an image registry.

**Procedure**

1. Build the image with the following command:

   ```bash
   $ docker build -t quay.io/my-repository/my-plugin:latest .
   ```

2. Optional: If you want to test your image, run the following command:

   ```bash
   $ docker run -it --rm -d -p 9001:80 quay.io/my-repository/my-plugin:latest
   ```

3. Push the image by running the following command:

   ```bash
   $ docker push quay.io/my-repository/my-plugin:latest
   ```

### 6.6. DEPLOY YOUR PLUG-IN ON A CLUSTER

After pushing an image with your changes to a registry, you can deploy the plug-in to a cluster.

**Procedure**

1. To deploy your plug-in to a cluster, instantiate the `template` by running the following command:

   ```bash
   $ oc process -f template.yaml
   ```
-p PLUGIN_NAME=my-plugin \ 1
-p NAMESPACE=my-plugin-namespace \ 2
-p IMAGE=quay.io/my-repository/my-plugin:latest \ 3
| oc create -f -

1. Update with the name of your plug-in.
2. Update with the namespace.
3. Update with the name of the image you created.

This command runs a light-weight NGINX HTTP server to serve the assets for your plug-in.

**IMPORTANT**

PLUGIN_NAME must match the plug-in name you used in the consolePlugin declaration of package.json.

2. Patch the Console Operator configuration to enable the plug-in by running the following command:

```bash
$ oc patch consoles.operator.openshift.io cluster --patch '{ "spec": { "plugins": ["my-plugin"] } }' --type=merge
```

6.7. ADDITIONAL RESOURCES

- Console customization plug-in.
- Console plug-in SDK README
- Dynamic plug-in enhancement proposal
CHAPTER 7. ABOUT THE DEVELOPER PERSPECTIVE IN THE WEB CONSOLE

The OpenShift Container Platform web console provides two perspectives; the Administrator perspective and the Developer perspective.

NOTE
The default web console perspective that is shown depends on the role of the user. The Developer perspective is displayed by default if the user is recognised as a developer.

The Developer perspective provides workflows specific to developer use cases, such as the ability to:

- Create and deploy applications on OpenShift Container Platform by importing existing codebases, images, and dockerfiles.
- Visually interact with applications, components, and services associated with them within a project and monitor their deployment and build status.
- Group components within an application and connect the components within and across applications.
- Integrate serverless capabilities (Technology Preview).
- Create workspaces to edit your application code using Eclipse Che.

7.1. PREREQUISITES
To access the Developer perspective, ensure that you have logged in to the web console.

7.2. ACCESSING THE DEVELOPER PERSPECTIVE
The Developer perspective in the OpenShift Container Platform web console provides workflows specific to developer use cases.

You can access the Developer perspective from the web console as follows:

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials. The default view for the OpenShift Container Platform web console is the Administrator perspective.

2. Use the perspective switcher to switch to the Developer perspective. The Topology view with a list of all the projects in your cluster is displayed.
3. Select an existing project from the list or use the **Project** drop-down list to create a new project.

If you have no workloads or applications in the project, the **Topology** view displays the available options to create applications. If you have existing workloads, the **Topology** view graphically displays your workload nodes.

**Additional resources**

- Creating and deploying applications on OpenShift Container Platform using the **Developer** perspective
- Viewing the applications in your project, verifying their deployment status, and interacting with them in the **Topology** view
CHAPTER 8. ABOUT THE WEB TERMINAL IN THE WEB CONSOLE

You can launch an embedded command line terminal instance in the web console. You must first install the Web Terminal Operator to use the web terminal.

NOTE

Cluster administrators can access the web terminal in OpenShift Container Platform 4.7 and later.

This terminal instance is preinstalled with common CLI tools for interacting with the cluster, such as oc, kubectl, odo, kn, tkn, helm, kubens, and kubectx. It also has the context of the project you are working on and automatically logs you in using your credentials.

IMPORTANT

Web terminal is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in 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 about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

8.1. INSTALLING THE WEB TERMINAL

You can install the web terminal using the Web Terminal Operator listed in the OpenShift Container Platform OperatorHub. When you install the Web Terminal Operator, the custom resource definitions (CRDs) that are required for the command line configuration, such as the DevWorkspace CRD, are automatically installed. The web console creates the required resources when you open the web terminal.

Prerequisites

- Access to an OpenShift Container Platform cluster using an account with cluster-admin permissions.

Procedure

1. In the Administrator perspective of the web console, navigate to Operators → OperatorHub.

2. Use the Filter by keyword box to search for the Web Terminal Operator in the catalog, and then click the Web Terminal tile.

3. Read the brief description about the Operator on the Web Terminal page, and then click Install.

4. On the Install Operator page, retain the default values for all fields.

   - The fast option in the Update Channel menu enables installation of the latest release of the Web Terminal Operator.
• The All namespaces on the cluster option in the Installation Mode menu enables the Operator to watch and be available to all namespaces in the cluster.

• The openshift-operators option in the Installed Namespace menu installs the Operator in the default openshift-operators namespace.

• The Automatic option in the Approval Strategy menu ensures that the future upgrades to the Operator are handled automatically by the Operator Lifecycle Manager.

5. Click Install.

6. In the Installed Operators page, click the View Operator to verify that the Operator is listed on the Installed Operators page.

NOTE
Prior to OpenShift Container Platform 4.8, the Web Terminal Operator bundled all its functionality in a single Operator. As of OpenShift Container Platform 4.8, the Web Terminal Operator installs the DevWorkspace Operator as a dependency to provide the same features.

7. After the Operator is installed, refresh your page to see the command line terminal icon on the upper right of the console.

8.2. USING THE WEB TERMINAL

After the Web Terminal Operator is installed, you can use the web terminal as follows:

1. To launch the web terminal, click the command line terminal icon ( ) on the upper right of the console. A web terminal instance is displayed in the Command line terminal pane. This instance is automatically logged in with your credentials.

2. Select the project where the DevWorkspace CR must be created from the Project drop-down list. By default, the current project is selected.

NOTE
• The DevWorkspace CR is created only if it does not already exist.
• The openshift-terminal project is the default project used for cluster administrators. They do not have the option to choose another project.

3. Click Start to initialize the web terminal using the selected project.

After the web terminal is initialized, you can use the preinstalled CLI tools like oc, kubectl, odo, kn, tkn, helm, kubens, and kubectx in the web terminal.

8.3. UNINSTALLING THE WEB TERMINAL

Uninstalling the web terminal is a two-step process:

1. Uninstall the Web Terminal Operator and related custom resources (CRs) that were added when you installed the Operator.
2. Uninstall the DevWorkspace Operator and its related custom resources that were added as a dependency of the Web Terminal Operator.

Uninstalling the Web Terminal Operator does not remove any of its custom resource definitions (CRDs) or managed resources that are created when the Operator is installed. These components must be manually uninstalled for security purposes. Removing these components also allows you to save cluster resources by ensuring that terminals do not idle when the Operator is uninstalled.

**Prerequisites**

- Access to an OpenShift Container Platform cluster using an account with `cluster-admin` permissions.

**8.3.1. Removing the Web Terminal Operator and the custom resources that support it**

Use the console and the CLI to delete any existing web terminals and CRs that were created during the installation of the Web Terminal Operator.

**NOTE**

Prior to OpenShift Container Platform 4.8, the Web Terminal Operator used different CRDs to provide Web Terminal capabilities. To uninstall versions 1.2.1 and below of the Web Terminal Operator, refer to the documentation for OpenShift Container Platform 4.7.

**Procedure**

1. Uninstall the Web Terminal Operator using the web console:
   
   a. In the **Administrator** perspective of the web console, navigate to **Operators → Installed Operators**.

   b. Scroll the filter list or type a keyword into the **Filter by name** box to find the **Web Terminal Operator**.

   c. Click the **Options menu** for the Web Terminal Operator, and then select **Uninstall Operator**.

   d. In the **Uninstall Operator** confirmation dialog box, click **Uninstall** to remove the Operator, Operator deployments, and pods from the cluster. The Operator stops running and no longer receives updates.

2. Remove the CRDs used by the Operator.

   ```
   $ oc delete devworkspaces.workspace.devfile.io --all-namespaces 
   --selector 'console.openshift.io/terminal=true' --wait
   
   $ oc delete devworkspacetemplates.workspace.devfile.io --all-namespaces 
   --selector 'console.openshift.io/terminal=true' --wait
   ```

**8.3.2. Deleting the DevWorkspace Operator dependency**
Use the CLI to delete the custom resource definitions (CRDs) and additional resources that are created during installation of the Web Terminal Operator.

**IMPORTANT**

The DevWorkspace Operator functions as a standalone Operator and may be required as a dependency for other Operators installed on the cluster (for example, the CodeReady Workspaces Operator may depend on it). Follow the steps below only if you are sure the DevWorkspace Operator is no longer needed.

**Procedure**

1. Remove the **DevWorkspace** custom resources used by the Operator, along with any related Kubernetes objects, such as deployments.

   ```bash
   $ oc delete devworkspaces.workspace.devfile.io --all-namespaces --all --wait
   $ oc delete devworkspaceroutings.controller.devfile.io --all-namespaces --all --wait
   ```

   **WARNING**
   
   If this step is not complete, finalizers make it difficult to fully uninstall the Operator easily.

2. Remove the CRDs used by the Operator:

   ```bash
   $ oc delete customresourcedefinitions.apiextensions.k8s.io devworkspaceroutings.controller.devfile.io
   $ oc delete customresourcedefinitions.apiextensions.k8s.io devworkspaces.workspace.devfile.io
   $ oc delete customresourcedefinitions.apiextensions.k8s.io devworkspacetemplates.workspace.devfile.io
   ```

   **WARNING**
   
   The DevWorkspace Operator provides custom resource definitions (CRDs) that use conversion webhooks. Failing to remove these CRDs can cause issues on the cluster.
$ oc delete customresourcedefinitions.apiextensions.k8s.io
devworkspaceoperatorconfigs.controller.devfile.io

3. Verify that all involved custom resource definitions are removed. The following command should not display any result.

$ oc get customresourcedefinitions.apiextensions.k8s.io | grep "devfile.io"

4. Remove the devworkspace-webhook-server deployment, mutating, and validating webhooks:

$ oc delete deployment/devworkspace-webhook-server -n openshift-operators
$ oc delete mutatingwebhookconfigurations controller.devfile.io
$ oc delete validatingwebhookconfigurations controller.devfile.io

**NOTE**

If you remove the devworkspace-webhook-server deployment without removing the mutating and validating webhooks, you will not be able to use oc exec commands to run commands in a container on the cluster. After you remove the webhooks you will be able to use the oc exec commands again.

5. Remove any remaining services, secrets, and config maps. Depending on the installation, some resources included in the following command may not exist on the cluster.

$ oc delete all --selector app.kubernetes.io/part-of=devworkspace-operator,app.kubernetes.io/name=devworkspace-webhook-server -n openshift-operators
$ oc delete serviceaccounts devworkspace-webhook-server -n openshift-operators
$ oc delete configmap devworkspace-controller -n openshift-operators
$ oc delete clusterrole devworkspace-webhook-server
$ oc delete clusterrolebinding devworkspace-webhook-server

6. Uninstall the Operator using the web console:

   a. In the Administrator perspective of the web console, navigate to Operators → Installed Operators.

   b. Scroll the filter list or type a keyword into the Filter by name box to find the DevWorkspace Operator.

   c. Click the Options menu for the DevWorkspace Operator, and then select Uninstall Operator.
d. In the **Uninstall Operator** confirmation dialog box, click **Uninstall** to remove the Operator, Operator deployments, and pods from the cluster. The Operator stops running and no longer receives updates.
CHAPTER 9. DISABLING THE WEB CONSOLE IN OPENSHIFT CONTAINER PLATFORM

You can disable the OpenShift Container Platform web console.

9.1. PREREQUISITES

- Deploy an OpenShift Container Platform cluster.

9.2. DISABLING THE WEB CONSOLE

You can disable the web console by editing the `consoles.operator.openshift.io` resource.

- Edit the `consoles.operator.openshift.io` resource:

  ```
  $ oc edit consoles.operator.openshift.io cluster
  ```

  The following example displays the parameters from this resource that you can modify:

  ```
  apiVersion: operator.openshift.io/v1
  kind: Console
  metadata:
    name: cluster
  spec:
    managementState: Removed
  ```

  Set the `managementState` parameter value to `Removed` to disable the web console. The other valid values for this parameter are `Managed`, which enables the console under the cluster’s control, and `Unmanaged`, which means that you are taking control of web console management.
CHAPTER 10. CREATING QUICK START TUTORIALS IN THE WEB CONSOLE

If you are creating quick start tutorials for the OpenShift Container Platform web console, follow these guidelines to maintain a consistent user experience across all quick starts.

10.1. UNDERSTANDING QUICK STARTS

A quick start is a guided tutorial with user tasks. In the web console, you can access quick starts under the Help menu. They are especially useful for getting oriented with an application, Operator, or other product offering.

A quick start primarily consists of tasks and steps. Each task has multiple steps, and each quick start has multiple tasks. For example:

- Task 1
  - Step 1
  - Step 2
  - Step 3

- Task 2
  - Step 1
  - Step 2
  - Step 3

- Task 3
  - Step 1
  - Step 2
  - Step 3

10.2. QUICK START USER WORKFLOW

When you interact with an existing quick start tutorial, this is the expected workflow experience:

1. In the Administrator or Developer perspective, click the Help icon and select Quick Starts.

2. Click a quick start card.

3. In the panel that appears, click Start.

4. Complete the on-screen instructions, then click Next.

5. In the Check your work module that appears, answer the question to confirm that you successfully completed the task.

   a. If you select Yes, click Next to continue to the next task.
b. If you select No, repeat the task instructions and check your work again.

6. Repeat steps 1 through 6 above to complete the remaining tasks in the quick start.

7. After completing the final task, click Close to close the quick start.

10.3. QUICK START COMPONENTS

A quick start consists of the following sections:

- **Card**: The catalog tile that provides the basic information of the quick start, including title, description, time commitment, and completion status
- **Introduction**: A brief overview of the goal and tasks of the quick start
- **Task headings**: Hyper-linked titles for each task in the quick start
- **Check your work module**: A module for a user to confirm that they completed a task successfully before advancing to the next task in the quick start
- **Hints**: An animation to help users identify specific areas of the product
- **Buttons**
  - **Next and back buttons**: Buttons for navigating the steps and modules within each task of a quick start
  - **Final screen buttons**: Buttons for closing the quick start, going back to previous tasks within the quick start, and viewing all quick starts

The main content area of a quick start includes the following sections:

- **Card copy**
- **Introduction**
- **Task steps**
- **Modals and in-app messaging**
- **Check your work module**

10.4. CONTRIBUTING QUICK STARTS

OpenShift Container Platform introduces the quick start custom resource, which is defined by a `ConsoleQuickStart` object. Operators and administrators can use this resource to contribute quick starts to the cluster.

**Prerequisites**

- You must have cluster administrator privileges.

**Procedure**

1. To create a new quick start, run:
1. Run:

```bash
$ oc get -o yaml consolequickstart spring-with-s2i > my-quick-start.yaml
```

2. Run:

```bash
$ oc create -f my-quick-start.yaml
```

3. Update the YAML file using the guidance outlined in this documentation.

4. Save your edits.

**10.4.1. Viewing the quick start API documentation**

**Procedure**

- To see the quick start API documentation, run:

```bash
$ oc explain consolequickstarts
```

Run `oc explain -h` for more information about `oc explain` usage.

**10.4.2. Mapping the elements in the quick start to the quick start CR**

This section helps you visually map parts of the quick start custom resource (CR) with where they appear in the quick start within the web console.

**10.4.2.1. conclusion element**

**Viewing the conclusion element in the YAML file**

```yaml
... summary:
  failed: Try the steps again.
  success: Your Spring application is running.
title: Run the Spring application
conclusion: >-
  Your Spring application is deployed and ready. 1
```

1 conclusion text

**Viewing the conclusion element in the web console**

The conclusion appears in the last section of the quick start.
10.4.2.2. description element

Viewing the description element in the YAML file

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleQuickStart
metadata:
  name: spring-with-s2i
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
...  
```

1 description text

Viewing the description element in the web console

The description appears on the introductory tile of the quick start on the Quick Starts page.
### 10.4.2.3. displayName element

**Viewing the displayName element in the YAML file**

```yaml
apiVersion: console.openshift.io/v1
kind: ConsoleQuickStart
metadata:
  name: spring-with-s2i
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
displayName: Get started with Spring
durationMinutes: 10
```

1. `displayName` text.

**Viewing the displayName element in the web console**

The display name appears on the introductory tile of the quick start on the Quick Starts page.
10.4.2.4. durationMinutes element

Viewing the durationMinutes element in the YAML file

```yaml
apiVersion: console.openshift.io/v1
class: ConsoleQuickStart
metadata:
  name: spring-with-s2i
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
displayName: Get started with Spring
durationMinutes: 10
```

The `durationMinutes` value, in minutes. This value defines how long the quick start should take to complete.

Viewing the durationMinutes element in the web console

The duration minutes element appears on the introductory tile of the quick start on the Quick Starts page.
10.4.2.5. icon element

Viewing the icon element in the YAML file

```
spec:
description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
displayName: Get started with Spring
durationMinutes: 10
icon:
  data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy5mcm9vdC9uZXQvb2EuY24iIiB4bWxuczp4bXQ6QWRvYmUgUGxhY2s=
```

OpenShift Container Platform 4.10 Web console

38
The icon defined as a base64 value.

Viewing the icon element in the web console

The icon appears on the introductory tile of the quick start on the Quick Starts page.
**Viewing the introduction element in the YAML file**

```
... introduction: >-
  **Spring** is a Java framework for building applications based on a distributed microservices architecture.

  - Spring enables easy packaging and configuration of Spring applications into a self-contained executable application which can be easily deployed as a container to OpenShift.

  - Spring applications can integrate OpenShift capabilities to provide a natural "Spring on OpenShift" developer experience for both existing and net-new Spring applications. For example:

    - Externalized configuration using Kubernetes ConfigMaps and integration with Spring Cloud Kubernetes

    - Service discovery using Kubernetes Services

    - Load balancing with Replication Controllers

    - Kubernetes health probes and integration with Spring Actuator

    - Metrics: Prometheus, Grafana, and integration with Spring Cloud Sleuth

    - Distributed tracing with Istio & Jaeger tracing

    - Developer tooling through Red Hat OpenShift and Red Hat CodeReady developer tooling to quickly scaffold new Spring projects, gain access to familiar Spring APIs in your favorite IDE, and deploy to Red Hat OpenShift
  ...
```

The introduction introduces the quick start and lists the tasks within it.

**Viewing the introduction element in the web console**

After clicking a quick start card, a side panel slides in that introduces the quick start and lists the tasks within it.
**Spring** is a Java framework for building applications based on a distributed microservices architecture.

- Spring enables easy packaging and configuration of Spring applications into a self-contained executable application which can be easily deployed as a container to OpenShift.
- Spring applications can integrate OpenShift capabilities to provide a natural "Spring on OpenShift" developer experience for both existing and net-new Spring applications. For example:
  - Externalized configuration using Kubernetes ConfigMaps and integration with Spring Cloud Kubernetes
  - Service discovery using Kubernetes Services
  - Load balancing with Replication Controllers
  - Kubernetes health probes and integration with Spring Actuator
  - Metrics: Prometheus, Grafana, and integration with Spring Cloud Sleuth
  - Distributed tracing with Istio & Jaeger tracing
  - Developer tooling through Red Hat OpenShift and Red Hat CodeReady developer tooling to quickly scaffold new Spring projects, gain access to familiar Spring APIs in your favorite IDE, and deploy to Red Hat OpenShift

In this quick start, you will complete 6 tasks:

1. Create a Spring application
2. View the build status
3. View the associated Git repository
4. View the pod status
5. Change the deployment icon to Spring
6. Run the Spring application

Start
10.4.3. Adding a custom icon to a quick start

A default icon is provided for all quick starts. You can provide your own custom icon.

Procedure

1. Find the .svg file that you want to use as your custom icon.

2. Use an online tool to convert the text to base64.

3. In the YAML file, add `icon: >-`, then on the next line include `data:image/svg+xml;base64` followed by the output from the base64 conversion. For example:

   ```yaml
   icon: >-
   data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHJvbGU9ImltZyIgdmlld.
   ```

10.4.4. Limiting access to a quick start

Not all quick starts should be available for everyone. The `accessReviewResources` section of the YAML file provides the ability to limit access to the quick start.

To only allow the user to access the quick start if they have the ability to create `HelmChartRepository` resources, use the following configuration:

```yaml
accessReviewResources:
  - group: helm.openshift.io
    resource: helmchartrepositories
    verb: create
```

To only allow the user to access the quick start if they have the ability to list Operator groups and package manifests, thus ability to install Operators, use the following configuration:

```yaml
accessReviewResources:
  - group: operators.coreos.com
    resource: operatorgroups
    verb: list
  - group: packages.operators.coreos.com
    resource: packagemanifests
    verb: list
```

10.4.5. Linking to other quick starts

Procedure

- In the `nextQuickStart` section of the YAML file, provide the `name`, not the `displayName`, of the quick start to which you want to link. For example:

  ```yaml
  nextQuickStart:
    - add-healthchecks
  ```
10.4.6. Supported tags for quick starts

Write your quick start content in markdown using these tags. The markdown is converted to HTML.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'b'</td>
<td>Defines bold text.</td>
</tr>
<tr>
<td>'img'</td>
<td>Embeds an image.</td>
</tr>
<tr>
<td>'i'</td>
<td>Defines italic text.</td>
</tr>
<tr>
<td>'strike'</td>
<td>Defines strike-through text.</td>
</tr>
<tr>
<td>'s'</td>
<td>Defines smaller text</td>
</tr>
<tr>
<td>'del'</td>
<td>Defines smaller text</td>
</tr>
<tr>
<td>'em'</td>
<td>Defines emphasized text</td>
</tr>
<tr>
<td>'strong'</td>
<td>Defines important text.</td>
</tr>
<tr>
<td>'a'</td>
<td>Defines an anchor tag.</td>
</tr>
<tr>
<td>'p'</td>
<td>Defines paragraph text.</td>
</tr>
<tr>
<td>'h1'</td>
<td>Defines a level 1 heading.</td>
</tr>
<tr>
<td>'h2'</td>
<td>Defines a level 2 heading.</td>
</tr>
<tr>
<td>'h3'</td>
<td>Defines a level 3 heading.</td>
</tr>
<tr>
<td>'h4'</td>
<td>Defines a level 4 heading.</td>
</tr>
<tr>
<td>'ul'</td>
<td>Defines an unordered list.</td>
</tr>
<tr>
<td>'ol'</td>
<td>Defines an ordered list.</td>
</tr>
<tr>
<td>'li'</td>
<td>Defines a list item.</td>
</tr>
<tr>
<td>'code'</td>
<td>Defines a text as code.</td>
</tr>
<tr>
<td>'pre'</td>
<td>Defines a block of preformatted text.</td>
</tr>
<tr>
<td>'button'</td>
<td>Defines a button in text.</td>
</tr>
</tbody>
</table>

10.4.7. Quick start highlighting markdown reference
The highlighting, or hint, feature enables Quick Starts to contain a link that can highlight and animate a component of the web console.

The markdown syntax contains:

- Bracketed link text
- The `highlight` keyword, followed by the ID of the element that you want to animate

### 10.4.7.1. Perspective switcher

```
[Perspective switcher]{{highlight qs-perspective-switcher}}
```

### 10.4.7.2. Administrator perspective navigation links

```
[Home]{{highlight qs-nav-home}}
[Operators]{{highlight qs-nav-operators}}
[Workloads]{{highlight qs-nav-workloads}}
[Serverless]{{highlight qs-nav-serverless}}
[Networking]{{highlight qs-nav-networking}}
[Storage]{{highlight qs-nav-storage}}
[Service catalog]{{highlight qs-nav-servicecatalog}}
[Compute]{{highlight qs-nav-compute}}
[User management]{{highlight qs-nav-usermanagement}}
[Administration]{{highlight qs-nav-administration}}
```

### 10.4.7.3. Developer perspective navigation links

```
[Add]{{highlight qs-nav-add}}
[Topology]{{highlight qs-nav-topology}}
[Search]{{highlight qs-nav-search}}
[Project]{{highlight qs-nav-project}}
[Helm]{{highlight qs-nav-helm}}
```

### 10.4.7.4. Common navigation links

```
[Builds]{{highlight qs-nav-builds}}
[Pipelines]{{highlight qs-nav-pipelines}}
[Monitoring]{{highlight qs-nav-monitoring}}
```

### 10.4.7.5. Masthead links

```
[CloudShell]{{highlight qs-masthead-cloudshell}}
[Utility Menu]{{highlight qs-masthead-utilitymenu}}
[User Menu]{{highlight qs-masthead-usermenu}}
[Applications]{{highlight qs-masthead-applications}}
[Import]{{highlight qs-masthead-import}}
[Help]{{highlight qs-masthead-help}}
[Notifications]{{highlight qs-masthead-notifications}}
```

### 10.4.8. Code snippet markdown reference
You can execute a CLI code snippet when it is included in a quick start from the web console. To use this feature, you must first install the Web Terminal Operator. The web terminal and code snippet actions that execute in the web terminal are not present if you do not install the Web Terminal Operator. Alternatively, you can copy a code snippet to the clipboard regardless of whether you have the Web Terminal Operator installed or not.

10.4.8.1. Syntax for inline code snippets

```
`code block`{{copy}}
`code block`{{execute}}
```

**NOTE**

If the `execute` syntax is used, the **Copy to clipboard** action is present whether you have the Web Terminal Operator installed or not.

10.4.8.2. Syntax for multi-line code snippets

```
```
```
 multi line code block
```
```
```
```

10.5. QUICK START CONTENT GUIDELINES

10.5.1. Card copy

You can customize the title and description on a quick start card, but you cannot customize the status.

- Keep your description to one to two sentences.
- Start with a verb and communicate the goal of the user. Correct example:

```
Create a serverless application.
```

10.5.2. Introduction

After clicking a quick start card, a side panel slides in that introduces the quick start and lists the tasks within it.

- Make your introduction content clear, concise, informative, and friendly.
- State the outcome of the quick start. A user should understand the purpose of the quick start before they begin.
- Give action to the user, not the quick start.
  - Correct example:
In this quick start, you will deploy a sample application to {product-title}.

- **Incorrect example:**

  This quick start shows you how to deploy a sample application to {product-title}.

- The introduction should be a maximum of four to five sentences, depending on the complexity of the feature. A long introduction can overwhelm the user.

- List the quick start tasks after the introduction content, and start each task with a verb. Do not specify the number of tasks because the copy would need to be updated every time a task is added or removed.

  - **Correct example:**

    Tasks to complete: Create a serverless application; Connect an event source; Force a new revision

  - **Incorrect example:**

    You will complete these 3 tasks: Creating a serverless application; Connecting an event source; Forcing a new revision

### 10.5.3. Task steps

After the user clicks **Start**, a series of steps appears that they must perform to complete the quick start.

Follow these general guidelines when writing task steps:

- Use “Click” for buttons and labels. Use “Select” for checkboxes, radio buttons, and drop-down menus.

- Use “Click” instead of "Click on"

  - **Correct example:**

    Click OK.

  - **Incorrect example:**

    Click on the OK button.

- Tell users how to navigate between **Administrator** and **Developer** perspectives. Even if you think a user might already be in the appropriate perspective, give them instructions on how to get there so that they are definitely where they need to be.

  Examples:

  - Enter the Developer perspective: In the main navigation, click the dropdown menu and select Developer.
  - Enter the Administrator perspective: In the main navigation, click the dropdown menu and select Admin.

- Use the “Location, action” structure. Tell a user where to go before telling them what to do.
○ Correct example:
  In the node.js deployment, hover over the icon.

○ Incorrect example:
  Hover over the icon in the node.js deployment.

- Keep your product terminology capitalization consistent.

- If you must specify a menu type or list as a dropdown, write “dropdown” as one word without a hyphen.

- Clearly distinguish between a user action and additional information on product functionality.
  ○ User action:
    Change the time range of the dashboard by clicking the dropdown menu and selecting time range.
  ○ Additional information:
    To look at data in a specific time frame, you can change the time range of the dashboard.

- Avoid directional language, like “In the top-right corner, click the icon”. Directional language becomes outdated every time UI layouts change. Also, a direction for desktop users might not be accurate for users with a different screen size. Instead, identify something using its name.
  ○ Correct example:
    In the navigation menu, click Settings.
  ○ Incorrect example:
    In the left-hand menu, click Settings.

- Do not identify items by color alone, like “Click the gray circle”. Color identifiers are not useful for sight-limited users, especially colorblind users. Instead, identify an item using its name or copy, like button copy.
  ○ Correct example:
    The success message indicates a connection.
  ○ Incorrect example:
    The message with a green icon indicates a connection.

- Use the second-person point of view, you, consistently:
  ○ Correct example:
    Set up your environment.
• Incorrect example:
  
  Let's set up our environment.

10.5.4. Check your work module

• After a user completes a step, a Check your work module appears. This module prompts the user to answer a yes or no question about the step results, which gives them the opportunity to review their work. For this module, you only need to write a single yes or no question.

• If the user answers Yes, a check mark will appear.

• If the user answers No, an error message appears with a link to relevant documentation, if necessary. The user then has the opportunity to go back and try again.

10.5.5. Formatting UI elements

Format UI elements using these guidelines:

• Copy for buttons, dropdowns, tabs, fields, and other UI controls: Write the copy as it appears in the UI and bold it.

• All other UI elements—including page, window, and panel names: Write the copy as it appears in the UI and bold it.

• Code or user-entered text: Use monospaced font.

• Hints: If a hint to a navigation or masthead element is included, style the text as you would a link.

• CLI commands: Use monospaced font.

• In running text, use a bold, monospaced font for a command.

• If a parameter or option is a variable value, use an italic monospaced font.

• Use a bold, monospaced font for the parameter and a monospaced font for the option.

10.6. ADDITIONAL RESOURCES

• For voice and tone requirements, refer to PatternFly’s brand voice and tone guidelines.

• For other UX content guidance, refer to all areas of PatternFly’s UX writing style guide.