



OpenShift Container Platform 4.10

Web コンソール

OpenShift Container Platform Web コンソールのスタートガイド

OpenShift Container Platform 4.10 Web コンソール

OpenShift Container Platform Web コンソールのスタートガイド

法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本書では、OpenShift Container Platform Web コンソールにアクセスし、カスタマイズする方法を説明します。

目次

第1章 WEB コンソールの概要	4
1.1. WEB コンソールの ADMINISTRATOR パースペクティブについて	4
1.2. WEB コンソールの DEVELOPER パースペクティブ	4
1.3. パースペクティブへのアクセス	5
第2章 WEB コンソールへのアクセス	7
2.1. 前提条件	7
2.2. WEB コンソールの理解および WEB コンソールへのアクセス	7
第3章 OPENSIFT CONTAINER PLATFORM DASHBOARD を使用したクラスター情報の取得	8
3.1. OPENSIFT CONTAINER PLATFORM ダッシュボードページについて	8
第4章 ユーザー設定の追加	10
4.1. ユーザー設定	10
第5章 OPENSIFT CONTAINER PLATFORM の WEB コンソールの設定	11
5.1. 前提条件	11
5.2. WEB コンソールの設定	11
5.3. WEB コンソールでのクイックスタートの無効化	11
第6章 OPENSIFT CONTAINER PLATFORM の WEB コンソールのカスタマイズ	13
6.1. カスタムロゴおよび製品名の追加	13
6.2. WEB コンソールでのカスタムリンクの作成	14
6.3. コンソールルートのカスタマイズ	15
6.4. ログインページのカスタマイズ	17
6.5. 外部ログリンクのテンプレートの定義	18
6.6. カスタム通知バナーの作成	19
6.7. CLI ダウンロードのカスタマイズ	19
6.8. YAML サンプルの KUBERNETES リソースへの追加	20
第7章 動的プラグインの OPENSIFT CONTAINER PLATFORM WEB コンソールへの追加	22
7.1. 動的プラグインについて	22
7.2. 動的プラグインを使い始める	23
7.3. DOCKER を使用したイメージのビルド	25
7.4. 動的プラグインの実行	25
7.5. クラスターへのプラグインのデプロイ	26
7.6. 関連情報	28
第8章 WEB 端末	29
8.1. WEB 端末のインストール	29
8.2. WEB 端末の使用	30
8.3. WEB 端末のトラブルシューティング	30
8.4. WEB 端末のアンインストール	31
第9章 OPENSIFT CONTAINER PLATFORM の WEB コンソールの無効化	35
9.1. 前提条件	35
9.2. WEB コンソールの無効化	35
第10章 WEB コンソールでのクイックスタートチュートリアル の作成	36
10.1. クイックスタートについて	36
10.2. クイックスタートのユーザーワークフロー	36
10.3. クイックスタートのコンポーネント	37
10.4. クイックスタートの継続	37
10.5. クイックスタートのコンテンツガイドライン	49

第1章 WEB コンソールの概要

Red Hat OpenShift Container Platform Web コンソールは、プロジェクトデータを視覚化し、管理およびトラブルシューティングタスクを実行するグラフィカルユーザーインターフェイスを提供します。Web コンソールは、`openshift-console` プロジェクトのコントロールプレーンノードで Pod として実行されます。これは **console-operator** Pod によって管理されます。**Administrator** および **Developer** パースペクティブの両方がサポートされます。

Administrator および **Developer** パースペクティブの両方で、OpenShift Container Platform のクイックスタートチュートリアルを作成できます。クイックスタートは、ユーザータスクに関するガイド付きチュートリアルで、アプリケーション、Operator、またはその他の製品オフリングを理解するのに役立ちます。

1.1. WEB コンソールの ADMINISTRATOR パースペクティブについて

Administrator パースペクティブでは、クラスターインベントリ、容量、全般的および特定の使用に関する情報、および重要なイベントのストリームを表示できます。これらはすべて、プランニングおよびトラブルシューティングの作業を簡素化するのに役立ちます。プロジェクト管理者とクラスター管理者の両方が **Administrator** パースペクティブを表示できます。

OpenShift Container Platform 4.7 以降の場合、クラスター管理者は Web Terminal Operator を使用して組み込みのコマンドラインターミナルインスタンスを開くこともできます。



注記

表示されるデフォルトの Web コンソールパースペクティブは、ユーザーのロールによって異なります。ユーザーが管理者として認識される場合、**Administrator** パースペクティブがデフォルトで表示されます。

Administrator パースペクティブは、以下を実行する機能などの管理者のユースケースに固有のワークフローを提供します。

- ワークロード、ストレージ、ネットワーク、およびクラスター設定を管理する。
- Operator Hub を使用して Operator をインストールし、管理する。
- ユーザーにログインを許可し、ロールおよびロールバインディングを使用してユーザーアクセスを管理できるようにするアイデンティティプロバイダーを追加する。
- クラスターの更新、部分的なクラスターの更新、クラスター Operator、カスタムリソース定義 (CRD)、ロールバインディング、リソースクォータなど、さまざまな高度な設定を表示および管理する。
- メトリクス、アラート、モニタリングダッシュボードなどのモニタリング機能にアクセスし、管理する。
- クラスターについてのロギング、メトリクス、および高ステータスの情報を表示し、管理する。
- OpenShift Container Platform の **Administrator** パースペクティブに関連するアプリケーション、コンポーネント、およびサービスと視覚的に対話する。

1.2. WEB コンソールの DEVELOPER パースペクティブ

Developer パースペクティブは、アプリケーション、サービス、データベースをデプロイするために組み込まれたさまざまな手法を提供します。**Developer** パースペクティブでは、以下を実行できます。

- コンポーネントでのロールアウトのローリングおよび再作成をリアルタイムに可視化する。
- アプリケーションのステータス、リソースの使用状況、プロジェクトイベントのストリーミング、およびクォータの消費を表示する。
- プロジェクトを他のユーザーと共有する。
- プロジェクトで Prometheus Query Language (PromQL) クエリーを実行し、グラフに可視化されたメトリクスを検査して、アプリケーションに関する問題のトラブルシューティングを行う。メトリクスにより、クラスターの状態と、モニターしているユーザー定義のワークロードに関する情報が提供されます。

OpenShift Container Platform 4.7 以降の場合、クラスター管理者は Web コンソールで組み込みのコマンドラインターミナルインスタンスを開くこともできます。



注記

表示されるデフォルトの Web コンソールパースペクティブは、ユーザーのロールによって異なります。**Developer** パースペクティブは、ユーザーが開発者として認識される場合、デフォルトで表示されます。

Developer パースペクティブは、以下を実行する機能を含む、開発者のユースケースに固有のワークフローを提供します。

- 既存のコードベース、イメージ、およびコンテナファイルをインポートして、OpenShift Container Platform でアプリケーションを作成し、デプロイします。
- アプリケーション、コンポーネント、およびプロジェクト内のこれらに関連付けられたサービスと視覚的に対話し、それらのデプロイメントとビルドステータスを監視します。
- アプリケーション内のコンポーネントをグループ化し、アプリケーション内およびアプリケーション間でコンポーネントを接続します。
- Serverless 機能 (テクノロジープレビュー) を統合します。
- Eclipse Che を使用してアプリケーションコードを編集するためのワークスペースを作成します。

Topology ビューを使用して、プロジェクトのアプリケーション、コンポーネント、およびワークロードを表示できます。プロジェクトにワークロードがない場合、**Topology** ビューにはワークロードを作成またはインポートするためのリンクがいくつか表示されます。**Quick Search** を使用してコンポーネントを直接インポートすることもできます。

関連情報

Developer パースペクティブで **Topology** ビューを使用する方法の詳細は、[Topology ビューを使用したアプリケーション設定の表示](#) を参照してください。

1.3. パースペクティブへのアクセス

次のように、Web コンソールから **Administrator** および **Developer** パースペクティブにアクセスできます。

前提条件

パースペクティブにアクセスするには、Web コンソールにログインしていることを確認してください。デフォルトのパースペクティブは、ユーザーの権限によって自動的に決定されます。すべてのプロジェクトへのアクセス権を持つユーザーには **Administrator** パースペクティブが選択され、自分のプロジェクトへのアクセスが制限されているユーザーには **Developer** パースペクティブが選択されます。

関連情報

パースペクティブの変更の詳細は、[ユーザー設定の追加](#) を参照してください。

手順

1. パースペクティブスイッチャーを使用して、**Administrator** パースペクティブまたは **Developer** パースペクティブに切り替えます。
2. **Project** ドロップダウンリストから既存のプロジェクトを選択します。このドロップダウンから新しいプロジェクトを作成することもできます。



注記

パースペクティブスイッチャーは、**cluster-admin** としてのみ使用できます。

関連情報

- [クラスター管理者について](#)
- [Administrator パースペクティブの概要](#)
- [Developer パースペクティブを使用して OpenShift Container Platform でアプリケーションを作成し、デプロイする](#)
- [Topology ビューを使用してプロジェクトにアプリケーションを表示し、デプロイメントのステータスを確認し、それらと対話する](#)
- [クラスター情報の表示](#)
- [Web コンソールの設定](#)
- [Web コンソールのカスタマイズ](#)
- [Web 端末の使用](#)
- [クイックスタートチュートリアルの作成](#)
- [Web コンソールの無効化](#)

第2章 WEB コンソールへのアクセス

OpenShift Container Platform Web コンソールは、Web ブラウザーからアクセスできるユーザーインターフェイスです。開発者は Web コンソールを使用してプロジェクトのコンテンツを視覚的に把握し、参照し、管理することができます。

2.1. 前提条件

- Web コンソールを使用するために JavaScript が有効にされている必要があります。WebSocket をサポートする Web ブラウザーを使用することが最も推奨されます。
- [OpenShift Container Platform 4.x のテスト済みインテグレーション](#) のページを確認してから、クラスターのサポートされるインフラストラクチャーを作成します。

2.2. WEB コンソールの理解および WEB コンソールへのアクセス

Web コンソールはマスター上で Pod として実行されます。Web コンソールを実行するために必要な静的アセットは Pod によって提供されます。OpenShift Container Platform が **openshift-install create cluster** を使用して正常にインストールされた後に、Web コンソールの URL およびインストールされたクラスターのログイン認証情報を、インストールプログラムの CLI 出力で確認します。以下に例を示します。

出力例

```
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>
```

これらの詳細を使用してログインし、Web コンソールにアクセスします。

インストールしていない既存のクラスターの場合、**oc whoami --show-console** を使用して Web コンソール URL を表示します。

関連情報

- [Web コンソールで機能セットの有効化](#)

第3章 OPENSIFT CONTAINER PLATFORM DASHBOARD を使用したクラスター情報の取得

OpenShift Container Platform Web コンソールから **Home** → **Dashboards** → **Overview** に移動し、クラスターの概要情報をキャプチャーする OpenShift Container Platform ダッシュボードにアクセスします。

OpenShift Container Platform ダッシュボードは、個別のダッシュボードカードでキャプチャーされるさまざまなクラスター情報を提供します。

3.1. OPENSIFT CONTAINER PLATFORM ダッシュボードページについて

OpenShift Container Platform ダッシュボードは以下のカードで設定されます。

- **Details** は、クラスターの詳細情報の概要を表示します。ステータスには、**ok**、**error**、**warning**、**in progress**、および **unknown** が含まれます。リソースでは、カスタムのステータス名を追加できます。
 - クラスター
 - プロバイダー
 - バージョン
- **Cluster Inventory** は、リソースの数および関連付けられたステータスの詳細を表示します。これは、問題の解決に介入が必要な場合に役立ちます。以下についての情報が含まれます。
 - ノード数
 - Pod 数
 - 永続ストレージボリューム要求
 - クラスター内のベアメタルホスト。これらはステータス別に一覧表示されます (**metal3** 環境でのみ利用可能)。
- **Cluster Capacity** チャートは、管理者が追加リソースがクラスターで必要になるタイミングを把握するのに役立ちます。このグラフには、現在の消費量を表示する内側の円が含まれ、外側の円には、以下の情報を含む、リソースに対して設定されたしきい値が表示されます。
 - CPU 時間
 - メモリー割り当て
 - 消費されるストレージ
 - 消費されるネットワークリソース
- **Cluster Utilization** は指定された期間における各種リソースの容量を表示します。これは、管理者がリソースの高い消費量の規模および頻度を理解するのに役立ちます。
- **Events** は、Pod の作成または別のホストへの仮想マシンの移行などのクラスター内の最近のアクティビティに関連したメッセージを一覧表示します。

- **Top Consumers** は、管理者がクラスターリソースの消費状況を把握するのに役立ちます。リソースをクリックし、指定されたクラスターリソース (CPU、メモリー、またはストレージ) の最大量を消費する Pod およびノードを一覧表示する詳細ページに切り替えます。

第4章 ユーザー設定の追加

要件に合わせてプロファイルのデフォルト設定を変更できます。デフォルトのプロジェクト、トポロジービュー (graph/list)、メディア (フォームまたは YAML) および言語設定を指定できます。

ユーザー設定への変更は自動的に保存されます。

4.1. ユーザー設定

クラスターのデフォルトのユーザー設定を指定できます。

手順

1. ログイン認証情報を使用して OpenShift Container Platform Web コンソールにログインします。
2. マストヘッドを使用して、ユーザープロファイルのユーザー名とパスワードにアクセスします。
3. **General** セクションで、以下を実行します。
 - a. **perspective** フィールドで、ログインするデフォルトのパースペクティブを設定できます。必要に応じて **Administrator** または **Developer** パースペクティブを選択できます。パースペクティブが選択されていない場合には、最後にアクセスしたパースペクティブにログインします。
 - b. **Project** フィールドで、作業するプロジェクトを選択します。コンソールは、毎回ログインすると、そのプロジェクトにデフォルト設定されます。
 - c. **Topology** フィールドで、トポロジービューのデフォルトをグラフビューか、一覧ビューに設定できます。選択されていない場合は、コンソールは使用した最後のビューにデフォルト設定されます。
 - d. **Create/Edit resource method** フィールドで、リソースの作成または編集設定を指定できます。フォームおよび YAML オプションの両方が利用可能な場合には、選択した内容にコンソールはデフォルト設定されます。
4. ブラウザーのデフォルトの言語設定を使用するには、言語セクションで、**Default browser language** を選択します。それ以外の場合は、コンソールに使用する言語を選択します。

第5章 OPENSIFT CONTAINER PLATFORM の WEB コンソールの設定

OpenShift Container Platform の Web コンソールを変更してログアウトリダイレクト URL を設定したり、コンソールを無効にしたりすることができます。

5.1. 前提条件

- OpenShift Container Platform クラスターをデプロイします。

5.2. WEB コンソールの設定

`console.config.openshift.io` リソースを編集して Web コンソールを設定できます。

- `console.config.openshift.io` リソースを編集します。

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

以下の例は、コンソールのリソース定義のサンプルを示しています。

```
apiVersion: config.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
  authentication:
    logoutRedirect: "" ①
status:
  consoleURL: "" ②
```

- ① ユーザーが Web コンソールからログアウトする際にロードするページの URL を指定します。値を指定しない場合、ユーザーは Web コンソールのログインページに戻ります。**logoutRedirect** URL を指定することにより、ユーザーはアイデンティティプロバイダー経由でシングルログアウト (SLO) を実行し、シングルサインオンセッションを破棄することができます。
- ② Web コンソール URL。これをカスタム値に更新するには、**Web コンソール URL のカスタマイズ**を参照してください。

5.3. WEB コンソールでのクイックスタートの無効化

Web コンソールの **Administrator** パースペクティブを使用して、1つ以上のクイックスタートを無効にできます。

前提条件

- クラスター管理者の権限があり、Web コンソールにログインしている。

手順

1. **Administrator** パースペクティブで、**Administration** → **Cluster Settings** に移動します。

2. **Cluster Settings** ページで、**Configuration** タブをクリックします。
3. **Configuration** ページで、**operator.openshift.io** と説明が記載されている **Console** 設定リソースをクリックします。

Cluster Settings

Details ClusterOperators **Configuration**

Edit the following resources to manage the configuration of your cluster.

console /

Configuration resource	Description	
Console config.openshift.io	Console holds cluster-wide configuration for the web console, including the logout URL, and reports the public URL of the console. The canonical name is 'cluster'. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).	⋮
Console operator.openshift.io	Console provides a means to configure an operator to manage the console. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).	⋮

4. **Action** ドロップダウンリストから **Customize** を選択し、**Cluster configuration** ページを開きます。
5. **General** タブの **Quick starts** セクションで、**Enabled** リストまたは **Disabled** リストから項目を選択し、矢印ボタンを使用して他方のリストに移動します。
 - 1つのクイックスタートを有効または無効にするには、該当するクイックスタートをクリックし、一重矢印ボタンを使用してクイックスタートを適切なリストに移動します。
 - 複数のクイックスタートをまとめて有効または無効にするには、Ctrl を押して移動するクイックスタートをクリックします。次に、一重矢印ボタンを使用してクイックスタートを適切なリストに移動します。
 - すべてのクイックスタートをまとめて有効または無効にするには、二重矢印ボタンをクリックして、すべてのクイックスタートを適切なリストに移動します。

第6章 OPENSIFT CONTAINER PLATFORM の WEB コンソールのカスタマイズ

OpenShift Container Platform の Web コンソールをカスタマイズして、カスタムロゴ、製品名、リンク、通知、およびコマンドラインのダウンロードを設定できます。これは、Web コンソールを企業や政府の特定要件を満たすように調整する必要がある場合にとくに役立ちます。

6.1. カスタムロゴおよび製品名の追加

カスタムロゴまたはカスタム製品名を追加することで、カスタムブランディングを作成できます。これらの設定は相互に独立しているため、両方またはいずれかを設定できます。

前提条件

- 管理者の権限があること。
- 使用するロゴのファイルを作成します。ロゴは、GIF、JPG、PNG、または SVG を含む共通のイメージ形式のファイルであり、**60px** の **max-height** に制限されます。

手順

1. ロゴファイルを **openshift-config** namespace の設定マップにインポートします。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

- ❶ 有効な base64 でエンコードされたロゴを指定します。

2. Web コンソールの Operator 設定を編集して、**customLogoFile** および **customProductName** を組み込みます。

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

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

```
key: console-custom-logo.png
name: console-custom-logo
customProductName: My Console
```

Operator 設定が更新されると、カスタムロゴ設定マップをコンソール namespace に同期し、これをコンソール Pod にマウントし、再デプロイします。

3. 正常に実行されたかどうかを確認します。問題がある場合は、コンソールクラスター Operator は **Degraded** ステータスを報告し、コンソール Operator 設定も **CustomLogoDegraded** ステータスを **KeyOrFilenameInvalid** または **NoImageProvided** などの理由と共に報告します。**clusteroperator** を確認するには、以下を実行します。

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

コンソール Operator 設定を確認するには、以下を実行します。

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

6.2. WEB コンソールでのカスタムリンクの作成

前提条件

- 管理者の権限があること。

手順

1. **Administration** → **Custom Resource Definitions** から、 **ConsoleLink** をクリックします。
2. **Instances** タブを選択します。
3. **Create Console Link** をクリックし、ファイルを編集します。

```
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: example
spec:
  href: 'https://www.example.com'
  location: HelpMenu ❶
  text: Link 1
```

- ❶ 有効な場所の設定は、**HelpMenu**、**UserMenu**、**ApplicationMenu**、および **NamespaceDashboard** です。

カスタムリンクがすべての namespace に表示されるようにするには、以下の例に従います。

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

```
href: 'https://www.example.com'
location: NamespaceDashboard
text: This appears in all namespaces
```

カスタムリンクが一部の namespace のみに表示されるようにするには、以下の例に従います。

```
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: namespaced-dashboard-for-some-namespaces
spec:
  href: 'https://www.example.com'
  location: NamespaceDashboard
  # This text will appear in a box called "Launcher" under "namespace" or "project" in the web
  console
  text: Custom Link Text
  namespaceDashboard:
    namespaces:
      # for these specific namespaces
      - my-namespace
      - your-namespace
      - other-namespace
```

カスタムリンクがアプリケーションメニューに表示されるようにするには、以下の例に従います。

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

4. **Save** をクリックして変更を適用します。

6.3. コンソールルートのカスタマイズ

console および **downloads** ルートについて、カスタムルート機能が **ingress** 設定ルート設定 API を使用します。**console** カスタムルートが **ingress** 設定と **console-operator** 設定の両方に設定されている場合、新規の **ingress** 設定のカスタムルート設定が優先されます。**console-operator** 設定を使用したルート設定は非推奨になりました。

6.3.1. コンソールルートのカスタマイズ

クラスター **Ingress** 設定の **spec.componentRoutes** フィールドにカスタムホスト名および TLS 証明書を設定して、コンソールルートのカスタマイズできます。

前提条件

- 管理者権限のあるユーザーでクラスターにログインしている。
- **openshift-config** namespace に TLS 証明書およびキーを含めたシークレットを作成している。これは、カスタムホスト名の接尾辞のドメインがクラスターのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

ヒント

oc create secret tls コマンドを使用して TLS シークレットを作成できます。

手順

1. クラスター **Ingress** 設定を編集します。

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

2. カスタムのホスト名を設定し、オプションで提供する証明書とキーを設定します。

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  componentRoutes:
  - name: console
    namespace: openshift-console
    hostname: <custom_hostname> ①
  servingCertKeyPairSecret:
    name: <secret_name> ②
```

- ① カスタムホスト名。
- ② TLS 証明書 (**tls.crt**) およびキー (**tls.key**) を含む **openshift-config** namespace のシークレットへの参照。これは、カスタムホスト名の接尾辞のドメインがクラスターのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

3. 変更を適用するためにファイルを保存します。

6.3.2. ダウンロードルートのカスタマイズ

クラスター **Ingress** 設定の **spec.componentRoutes** フィールドにカスタムホスト名および TLS 証明書を設定して、ダウンロードルートをカスタマイズできます。

前提条件

- 管理者権限のあるユーザーでクラスターにログインしている。
- **openshift-config** namespace に TLS 証明書およびキーを含めたシークレットを作成している。これは、カスタムホスト名の接尾辞のドメインがクラスターのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

ヒント

`oc create secret tls` コマンドを使用して TLS シークレットを作成できます。

手順

1. クラスタ **Ingress** 設定を編集します。

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

2. カスタムのホスト名を設定し、オプションで提供する証明書とキーを設定します。

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  componentRoutes:
    - name: downloads
      namespace: openshift-console
      hostname: <custom_hostname> ①
  servingCertKeyPairSecret:
    name: <secret_name> ②
```

- ① カスタムホスト名。
- ② TLS 証明書 (**tls.crt**) およびキー (**tls.key**) を含む **openshift-config** namespace のシークレットへの参照。これは、カスタムホスト名の接尾辞のドメインがクラスタのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

3. 変更を適用するためにファイルを保存します。

6.4. ログインページのカスタマイズ

サービス利用規約情報をカスタムログインページを使用して作成します。カスタムログインページは、GitHub や Google などのサードパーティーログインプロバイダーを使用している場合にも、ユーザーが信頼し、予想できるブランドのページを提示して、その後にユーザーを認証プロバイダーにリダイレクトする際に役立ちます。また、認証プロセス中にカスタムエラーページをレンダリングすることもできます。



注記

エラーテンプレートのカスタマイズは、要求ヘッダーや OIDC ベースの IDP などのリダイレクトを使用するアイデンティティプロバイダー (IDP) に限定されます。LDAP や htpasswd などのダイレクトパスワード認証を使用する IDP にはこれによる影響がありません。

前提条件

- 管理者の権限があること。

手順

1. 以下のコマンドを実行して、変更可能なテンプレートを作成します。

```
$ oc adm create-login-template > login.html
```

```
$ oc adm create-provider-selection-template > providers.html
```

```
$ oc adm create-error-template > errors.html
```

2. シークレットを作成します。

```
$ oc create secret generic login-template --from-file=login.html -n openshift-config
```

```
$ oc create secret generic providers-template --from-file=providers.html -n openshift-config
```

```
$ oc create secret generic error-template --from-file=errors.html -n openshift-config
```

3. 以下を実行します。

```
$ oc edit oauths cluster
```

4. 仕様を更新します。

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

oc explain oauths.spec.templates を実行して、オプションを把握します。

6.5. 外部ログリンクのテンプレートの定義

ログの参照に役立つサービスに接続しているものの、特定の 방법으로 URL を生成する必要がある場合は、リンクのテンプレートを定義できます。

前提条件

- 管理者の権限があること。

手順

1. **Administration** → **Custom Resource Definitions** から、 **ConsoleExternalLogLink** をクリックします。
2. **Instances** タブを選択します。
3. **Create Console External Log Link** をクリックし、ファイルを編集します。

```
apiVersion: console.openshift.io/v1
```

```

kind: ConsoleExternalLogLink
metadata:
  name: example
spec:
  hrefTemplate: >-
    https://example.com/logs?
resourceName=${resourceName}&containerName=${containerName}&resourceNamespace=${
resourceNamespace}&podLabels=${podLabels}
text: Example Logs

```

6.6. カスタム通知バナーの作成

前提条件

- 管理者の権限があること。

手順

1. Administration → Custom Resource Definitions から、ConsoleNotification をクリックします。
2. Instances タブを選択します。
3. Create Console Notification をクリックし、ファイルを編集します。

```

apiVersion: console.openshift.io/v1
kind: ConsoleNotification
metadata:
  name: example
spec:
  text: This is an example notification message with an optional link.
  location: BannerTop 1
  link:
    href: 'https://www.example.com'
    text: Optional link text
  color: '#fff'
  backgroundColor: '#0088ce'

```

- 1** 有効な場所の設定は、**BannerTop**、**BannerBottom**、および **BannerTopBottom** です。

4. Create をクリックして変更を適用します。

6.7. CLI ダウンロードのカスタマイズ

ファイルパッケージを直接ポイントしたり、パッケージを提供する外部ページをポイントできるカスタムのリンクテキストおよび URL を使用して、CLI をダウンロードするリンクを設定できます。

前提条件

- 管理者の権限があること。

手順

1. **Administration** → **Custom Resource Definitions**に移動します。
2. カスタムリソース定義 (CRD) の一覧から **ConsoleCLIDownload** を選択します。
3. **YAML** タブをクリックし、編集を行います。

```
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:
  - href: 'https://www.example.com/public/foo.tar'
    text: foo for linux
  - href: 'https://www.example.com/public/foo.mac.zip'
    text: foo for mac
  - href: 'https://www.example.com/public/foo.win.zip'
    text: foo for windows
```

4. **Save** ボタンをクリックします。

6.8. YAML サンプルの KUBERNETES リソースへの追加

YAML サンプルはいつでも Kubernetes リソースに動的に追加できます。

前提条件

- クラスタ管理者の権限があること。

手順

1. **Administration** → **Custom Resource Definitions**から、**ConsoleYAMLSample** をクリックします。
2. **YAML** をクリックし、ファイルを編集します。

```
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:
      - name: counter
        image: centos:7
        command:
          - "bin/bash"
          - "-c"
          - "for i in 9 8 7 6 5 4 3 2 1 ; do echo $i ; done"
    restartPolicy: Never
```

spec.snippet を使用して、YAML サンプルが完全な YAML リソース定義ではなく、ユーザーのカーソルで既存の YAML ドキュメントに挿入できる断片を示します。

3. **Save** をクリックします。

第7章 動的プラグインの OPENSIFT CONTAINER PLATFORM WEB コンソールへの追加

ランタイム時に読み込まれるクラスターに動的プラグインを作成し、デプロイできます。



重要

動的プラグインの作成は、テクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

7.1. 動的プラグインについて

動的プラグインを使用すると、実行時にカスタムページおよびその他のエクステンションをインターフェイスに追加できます。**ConsolePlugin** カスタムリソースはコンソールと共にプラグインを登録し、クラスター管理者は **console-operator** 設定でプラグインを有効にします。

7.1.1. 主な特長

動的プラグインを使用すると、以下のカスタマイズを OpenShift Container Platform エクスペリエンスに設定することができます。

- カスタムページの追加。
- 管理者と開発者を越えたパースペクティブを追加します。
- ナビゲーション項目の追加。
- リソースページへのタブおよびアクションの追加。

7.1.2. 全般的なガイドライン

プラグインの作成時には、以下の一般的なガイドラインに従ってください。

- プラグインをビルドして実行するには、**Node.js** と **yarn** が必要です。
- CSS クラス名の前にプラグイン名を付けて、競合を回避します。例: **my-plugin__heading** および **my-plugin__icon**
- 他のコンソールページとの一貫したルック、フィールド、および動作を維持します。
- プラグインの作成時には、**react-i18next** のローカリゼーションガイドラインに従ってください。以下の例のように **useTranslation** フックを使用できます。

```
const Header: React.FC = () => {
  const { t } = useTranslation('plugin__console-demo-plugin');
  return <h1>{t('Hello, World!')}</h1>;
};
```

- 要素セレクターなど、プラグインコンポーネント外のマークアップに影響を与える可能性のあるセレクターは避けてください。これらは API ではなく、変更される可能性があります。これらを使用すると、プラグインが破損する可能性があります。

PatternFly 4 ガイドライン

プラグインを作成する場合は、PatternFly の使用に関する以下のガイドラインに従ってください。

- [PatternFly4](#) コンポーネントおよび PatternFly CSS 変数を使用します。コア PatternFly コンポーネントは SDK から利用できます。Pattern Fly コンポーネントと変数を使用すると、将来のコンソールバージョンでプラグインが一貫しているように見えます。
- [PatternFly のアクセシビリティの基本](#) に従って、プラグインにアクセスできるようにします。
- Bootstrap や Tailwind などの他の CSS ライブラリーは使用しないでください。これらは、PatternFly と競合する可能性があり、コンソールのルックアンドフィールとは一致しません。

7.2. 動的プラグインを使い始める

OpenShift Container Platform Web コンソールに対して行うことができるさまざまなカスタマイズがあります。新しい OpenShift Console 動的プラグインを作成するように環境をセットアップし、プラグインの拡張例として **Pod details** ページにタブを追加します。



注記

OpenShift Container Platform Web コンソールは、ログインしているクラスターに接続されているコンテナで実行されます。独自のプラグインを作成する前にプラグインをテストする方法は、「動的プラグインの実行」を参照してください。

前提条件

- [Node.js](#) がインストールされていることを確認します。
- [yarn](#) がインストールされていることを確認します。

手順

1. 新しいタブで、[console-plugin-template](#) リポジトリを開きます。このリポジトリには、新しいタブでプラグインを作成するためのテンプレートが含まれています。



重要

カスタムプラグインコードは、Red Hat ではサポートされていません。プラグインで利用できるのは、[共同コミュニティのサポート](#) のみです。

2. **Use this template** → **Create new repository** をクリックして、テンプレートの GitHub リポジトリを作成します。
3. プラグインの名前で新しいリポジトリの名前を変更します。
4. コードを編集できるように、新しいリポジトリのクローンをローカルマシンに作成します。
5. **package.json** ファイルを編集して、プラグインのメタデータを **consolePlugin** 宣言に追加します。以下に例を示します。

■

```
"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": "/*"
  }
}
```

- ① プラグインの名前を更新します。
- ② バージョンを更新します。
- ③ プラグインの表示名を更新します。
- ④ プラグインの概要を使用して、説明を更新します。

6. **console-extensions.json** ファイルに以下を追加します。

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

7. **package.json** ファイルを編集して以下の変更を追加します。

```
"exposedModules": {
  "ExamplePage": "./components/ExamplePage",
  "ExampleTab": "./components/ExampleTab"
}
```

8. 新しいファイル **src/components/ExampleTab.tsx** を作成し、以下のスクリプトを追加することで、**Pod** ページの新規カスタムタブに表示されるメッセージを作成します。

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

export default function ExampleTab() {
  return (
```

```

    <p>This is a custom tab added to a resource using a dynamic plugin.</p>
  );
}

```

9. プラグインをクラスターにデプロイするには、プラグインの名前を Helm リリース名として Helm チャートを、新しい namespace または **-n** コマンドラインオプションで指定された既存の namespace にインストールします。次のコマンドを使用して、**plugin.image** パラメーター内のイメージの場所を指定します。

```

$ helm upgrade -i my-plugin charts/openshift-console-plugin -n my-plugin-namespace --
create-namespace --set plugin.image=my-plugin-image-location

```



注記

クラスターへのプラグインのデプロイの詳細は、「クラスターへのプラグインのデプロイ」を参照してください。

検証

- Pod ページに移動し、追加されたタブを表示します。

7.3. DOCKER を使用したイメージのビルド

クラスターにプラグインをデプロイするには、イメージをビルドし、これをイメージレジストリーにプッシュする必要があります。

手順

1. 以下のコマンドでイメージをビルドします。

```

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

```

2. オプション: イメージをテストする場合は、以下のコマンドを実行します。

```

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

```

3. 以下のコマンドを実行してイメージをプッシュします。

```

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

```

7.4. 動的プラグインの実行

ローカルの開発環境を使用してプラグインを実行できます。OpenShift コンソールは、ログインしているクラスターに接続されているコンテナで実行されます。

前提条件

- OpenShift CLI (**oc**) がインストールされている必要があります。
- OpenShift クラスターが実行中である必要があります。
- **Docker** または少なくとも v3.2.0 の **Podman** がインストールされている必要があります。

手順

- クローンを作成したリポジトリのローカルディレクトリーで2つのターミナルウィンドウを開きます。
 - 1つ目のターミナルで以下のコマンドを実行します。

```
$ yarn install
```

```
$ yarn run start
```

- 2つ目のターミナルウィンドウで次のコマンドを実行します。

```
$ oc login
```

```
$ yarn run start-console
```

検証

- [ローカルホスト](#) にアクセスして、実行中のプラグインを表示します。

7.5. クラスターへのプラグインのデプロイ

レジストリーに変更を加えたイメージをプッシュした後、プラグインをクラスターにデプロイできます。

手順

- プラグインをクラスターにデプロイするには、プラグインの名前を Helm リリース名として Helm チャートを、新しい namespace または **-n** コマンドラインオプションで指定された既存の namespace にインストールします。次のコマンドを使用して、**plugin.image** パラメーター内のイメージの場所を指定します。

```
$ helm upgrade -i my-plugin charts/openshift-console-plugin -n my-plugin-namespace --create-namespace --set plugin.image=my-plugin-image-location
```

ここでは、以下ようになります。

n <my-plugin-namespace>

プラグインをデプロイする既存の namespace を指定します。

--create-namespace

オプション: 新しい namespace にデプロイする場合は、このパラメーターを使用します。

--set plugin.image=my-plugin-image-location

plugin.image パラメーター内のイメージの場所を指定します。

- オプション: **charts/openshift-console-plugin/values.yaml** ファイルでサポートされている一連のパラメーターを使用して、追加のパラメーターを指定できます。

```
plugin:
  name: ""
  description: ""
  image: ""
```

```
imagePullPolicy: IfNotPresent
replicas: 2
port: 9443
securityContext:
  enabled: true
podSecurityContext:
  enabled: true
  runAsNonRoot: true
  seccompProfile:
    type: RuntimeDefault
containerSecurityContext:
  enabled: true
  allowPrivilegeEscalation: false
  capabilities:
    drop:
      - ALL
resources:
  requests:
    cpu: 10m
    memory: 50Mi
basePath: /
certificateSecretName: ""
serviceAccount:
  create: true
  annotations: {}
  name: ""
patcherServiceAccount:
  create: true
  annotations: {}
  name: ""
jobs:
  patchConsoles:
    enabled: true
    image: "registry.redhat.io/openshift4/ose-tools-
rhel8@sha256:e44074f21e0cca6464e50cb6ff934747e0bd11162ea01d522433a1a1ae116103"

  podSecurityContext:
    enabled: true
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containerSecurityContext:
    enabled: true
    allowPrivilegeEscalation: false
    capabilities:
      drop:
        - ALL
  resources:
    requests:
      cpu: 10m
      memory: 50Mi
```

検証

- Overview ページで、または Administration → Cluster Settings → Configuration → Console operator.openshift.io → Console plugins に移動して、有効なプラグインのリストを表示できます。



注記

新しいプラグイン設定が表示されるまで数分かかる場合があります。最近プラグインを有効にしたにもかかわらず、プラグインが表示されない場合は、ブラウザを更新する必要があります。実行時にエラーが発生した場合は、ブラウザ開発者ツールの JS コンソールをチェックして、プラグインコードにエラーがないか調べてください。

7.6. 関連情報

- [Helm について](#)

第8章 WEB 端末

8.1. WEB 端末のインストール

OpenShift Container Platform OperatorHub に一覧表示されている Web Terminal Operator を使用して Web 端末をインストールできます。Web 端末 Operator をインストールする際に、**DevWorkspace** CRD などのコマンドラインの設定に必要なカスタムリソース定義 (CRD) が自動的にインストールされます。Web コンソールでは、Web 端末を開く際に必要なリソースを作成します。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- クラスター管理者パーミッションがある。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → OperatorHub** に移動します。
2. **Filter by keyword** ボックスを使用してカタログで Web Terminal Operator を検索し、**Web Terminal** タイルをクリックします。
3. **Web Terminal** ページで Operator についての簡単な説明を確認してから、**Install** をクリックします。
4. **Install Operator** ページで、すべてのフィールドのデフォルト値を保持します。
 - **Update Channel** メニューの **fast** オプションは、Web 端末 Operator の最新リリースのインストールを可能にします。
 - **Installation Mode** メニューの **All namespaces on the cluster** オプションにより、Operator にクラスターのすべての namespace を監視され、Operator をこれらの namespace で利用可能にすることができます。
 - **Installed Namespace** メニューの **openshift-operators** オプションは、Operator をデフォルトの **openshift-operators** namespace にインストールします。
 - **Approval Strategy** メニューの **Automatic** オプションにより、Operator への今後のアップグレードは Operator Lifecycle Manager によって自動的に処理されます。
5. **Install** をクリックします。
6. **Installed Operators** ページで、**View Operator** をクリックし、Operator が **Installed Operators** ページに一覧表示されていることを確認します。



注記

Web Terminal Operator は、DevWorkspace Operator を依存関係としてインストールします。

7. Operator のインストール後に、ページを更新し、コンソールのマストヘッドにあるコマンドラインインターナルアイコン () を確認します。

8.2. WEB 端末の使用

Web コンソールで組み込みコマンドラインターミナルインスタンスを起動できます。この端末のインスタンスは、**oc**、**kubectrl**、**odo**、**kn**、**tkn**、**helm**、**kubens**、**subctl** および **kubectx** などのクラスターと対話するための一般的な CLI ツールと共に事前にインストールされます。また、これには作業しているプロジェクトのコンテキストが含まれ、ユーザーの認証情報を使用してユーザーのログインを自動的に行います。

8.2.1. Web 端末へのアクセス

Web Terminal Operator をインストールすると、Web 端末にアクセスできます。ターミナルで実行したコマンドの一覧からコマンドを選択して、コマンドを再実行することができます。これらのコマンドは、複数のターミナルセッション間で保持されます。Web 端末を閉じるまで、またはブラウザーウィンドウかタブを閉じるまで、Web 端末は表示されたままになります。

前提条件

- OpenShift Container Platform クラスターにアクセスでき、Web コンソールにログインしている。
- Web Terminal Operator がクラスターにインストールされている。

手順

1. Web 端末を起動するには、コンソールのマストヘッドにあるコマンドラインターミナルアイコン () をクリックします。Web 端末インスタンスが、**Command line terminal** ペインに表示されます。このインスタンスは、お使いの認証情報を使用して自動的にログインします。
2. **Project** ドロップダウンリストから **DevWorkspace** CR を作成する必要があるプロジェクトを選択します。デフォルトでは、現在のプロジェクトが選択されます。



注記

- **DevWorkspace** CR は存在しない場合にのみ作成されます。
 - **openshift-terminal** プロジェクトは、クラスター管理者に使用されるデフォルトのプロジェクトです。別のプロジェクトを選択するオプションはありません。
3. **Start** をクリックし、選択したプロジェクトを使用して Web 端末を初期化します。Web 端末を初期化した後に、Web 端末で **oc**、**kubectrl**、**odo**、**kn**、**tkn**、**helm**、**kubens**、**subctl**、および **kubectx** などの事前インストールされた CLI ツールを使用できます。

8.3. WEB 端末のトラブルシューティング

8.3.1. Web 端末とネットワークポリシー

クラスターにネットワークポリシーが設定されている場合、Web 端末の起動に失敗する可能性があります。Web 端末インスタンスを初期化するには、Web Terminal Operator は Web 端末の Pod と通信して実行中であることを確認する必要があります。OpenShift Container Platform Web コンソールはターミナル内のクラスターへの自動ログイン情報を送信する必要があります。いずれかのステップに失敗した場合には、Web 端末は初期化に失敗し、端末パネルはロード状態にあるように見えます。

この問題を回避するには、端末に使用される namespace のネットワークポリシーが **openshift-console** および **openshift-operators** namespace からの ingress を許可していることを確認してください。

8.4. WEB 端末のアンインストール

Web Terminal Operator をアンインストールしても、Operator のインストール時に作成されるカスタムリソース定義 (CRD) または管理リソースは削除されません。セキュリティ上の理由から、これらのコンポーネントは手動でアンインストールする必要があります。これらのコンポーネントを削除すると、Operator をアンインストールしても端末はアイドル状態にならないため、クラスターリソースが保存されます。

Web 端末のアンインストールは 2 つの手順で実行されます。

1. Operator のインストール時に追加された Web 端末 Operator および関連するカスタムリソース (CR) をアンインストールします。
2. Web 端末 Operator の依存関係として追加された DevWorkspace Operator とそれに関連するカスタムリソースをアンインストールします。

8.4.1. Web Terminal Operator の削除

Web 端末をアンインストールするには、Operator が使用する Web Terminal Operator とカスタムリソースを削除します。

前提条件

- クラスター管理者のパーミッションを持つ OpenShift Container Platform クラスターにアクセスできる。
- **oc** CLI がインストールされている。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. フィルターリストをスクロールするか、**Filter by name** ボックスにキーワードを入力して Web Terminal Operator を見つけます。
3. Web Terminal Operator の Options メニュー  をクリックし、**Uninstall Operator** を選択します。
4. **Uninstall Operator** 確認ダイアログボックスで、**Uninstall** をクリックし、Operator、Operator デプロイメント、および Pod をクラスターから削除します。この Operator は実行を停止し、更新を受信しなくなります。
5. カスタムリソースを削除します。

```
$ 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
```

8.4.2. DevWorkspace Operator の削除

Web 端末を完全にアンインストールするには、Operator が使用する DevWorkspace Operator とカスタムリソースも削除する必要があります。



重要

DevWorkspace Operator はスタンドアロン Operator であり、クラスターにインストールされている他の Operator の依存関係として必要になる場合があります。DevWorkspace Operator が不要であることが確実な場合にのみ、以下の手順を実行してください。

前提条件

- クラスター管理者のパーミッションを持つ OpenShift Container Platform クラスターにアクセスできる。
- **oc** CLI がインストールされている。

手順

1. Operator が使用する **DevWorkspace** カスタムリソースと関連する Kubernetes オブジェクトを削除します。

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

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



警告

この手順が完了していない場合、ファイナライザーにより Operator を完全にアンインストールすることが困難になります。

2. Operator によって使用される CRD を削除します。



警告

DevWorkspace Operator は、変換 Webhook を使用するカスタムリソース定義 (CRD) を提供します。これらの CRD の削除に失敗すると、クラスターで問題が発生する可能性があります。

```
$ 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
```

```
$ oc delete customresourcedefinitions.apiextensions.k8s.io
devworkspaceoperatorconfigs.controller.devfile.io
```

3. 関連するすべてのカスタムリソース定義が削除されていることを確認します。以下のコマンドを実行しても何も出力されないはずです。

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

4. **devworkspace-webhook-server** デプロイメント、変更用および検証用の Webhook を削除します。

```
$ oc delete deployment/devworkspace-webhook-server -n openshift-operators
```

```
$ oc delete mutatingwebhookconfigurations controller.devfile.io
```

```
$ oc delete validatingwebhookconfigurations controller.devfile.io
```



注記

変更用および検証用の Webhook を削除せずに **devworkspace-webhook-server** デプロイメントを削除した場合、**oc exec** コマンドを使用してクラスタのコンテナでコマンドを実行できません。Webhook を削除したら、**oc exec** コマンドを再度使用できます。

5. 残りのサービス、シークレット、および設定マップを削除します。インストールによっては、以下のコマンドに含まれる一部のリソースがクラスタに存在しない場合があります。

```
$ 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 clusterrole devworkspace-webhook-server
```

```
$ oc delete clusterrolebinding devworkspace-webhook-server
```

6. DevWorkspace Operator をアンインストールします。

- a. Web コンソールの **Administrator** パースペクティブで、**Operators** → **Installed Operators** に移動します。

- b. フィルターリストをスクロールするか、**Filter by name** ボックスにキーワードを入力して DevWorkspace Operator を見つけます。
- c. Operator のオプションメニュー  をクリックし、**Uninstall Operator** を選択します。
- d. **Uninstall Operator** 確認ダイアログボックスで、**Uninstall** をクリックし、Operator、Operator デプロイメント、および Pod をクラスターから削除します。この Operator は実行を停止し、更新を受信しなくなります。

第9章 OPENSIFT CONTAINER PLATFORM の WEB コンソールの無効化

OpenShift Container Platform の Web コンソールを無効にすることができます。

9.1. 前提条件

- OpenShift Container Platform クラスターをデプロイします。

9.2. WEB コンソールの無効化

`consoles.operator.openshift.io` リソースを編集して Web コンソールを無効にすることができます。

- `consoles.operator.openshift.io` リソースを編集します。

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

以下の例は、変更できるリソースのパラメーターを表示しています。

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

- ❶ **managementState** パラメーター値を **Removed** に設定し、Web コンソールを無効にします。このパラメーターの他の有効な値には以下が含まれます。**Managed** ではクラスターの制御下でコンソールを有効にし、**Unmanaged** は Web コンソール管理を制御するのがユーザーであることを意味します。

第10章 WEB コンソールでのクイックスタートチュートリアルの作成

OpenShift Container Platform Web コンソールのクイックスタートチュートリアルを作成する場合は、以下のガイドラインに従って、すべてのクイックスタートで一貫したユーザーエクスペリエンスを維持するようにしてください。

10.1. クイックスタートについて

クイックスタートは、ユーザータスクに関するガイド付きチュートリアルです。Web コンソールでは、**Help** メニューでクイックスタートにアクセスできます。これらは、アプリケーション、Operator、または他の製品オファリングを使用する場合に役立ちます。

クイックスタートは、主にタスクとステップで設定されます。タスクごとに複数のステップがあり、各クイックスタートには複数のタスクがあります。以下に例を示します。

- タスク 1
 - ステップ 1
 - ステップ 2
 - ステップ 3
- タスク 2
 - ステップ 1
 - ステップ 2
 - ステップ 3
- タスク 3
 - ステップ 1
 - ステップ 2
 - ステップ 3

10.2. クイックスタートのユーザーワークフロー

既存のクイックスタートチュートリアルと対話する場合、以下が想定されるワークフローエクスペリエンスになります。

1. **Administrator** または **Developer** パースペクティブで、**Help** アイコン をクリックし、**Quick Starts** を選択します。
2. クイックスタートカードをクリックします。
3. 表示されるパネルで **Start** をクリックします。
4. 画面上の手順を実行し、**Next** をクリックします。
5. 表示される **Check your work** モジュールで質問に回答し、タスクが正常に完了したことを確認します。

- a. **Yes** を選択した場合には、**Next** をクリックして次のタスクに進みます。
 - b. **No** を選択した場合は、タスクの手順を繰り返して作業を再度確認します。
6. 上記の手順1から6を繰り返し、クイックスタートの残りのタスクを完了します。
 7. 最終タスクが完了したら、**Close** をクリックしてクイックスタートを閉じます。

10.3. クイックスタートのコンポーネント

クイックスタートは、以下のセクションで設定されます。

- **Card**: タイトル、説明、時間 (time commitment)、完了ステータスなどの、クイックスタートの基本情報を提供するカタログタイトル
- **Introduction**: クイックスタートの目的およびタスクの概要
- **Task headings**: クイックスタートの各タスクのハイパーリンクタイトル
- **Check your work module** ユーザーがクイックスタートの次のタスクに進む前に、タスクが正常に完了したことを確認するためのモジュール
- **Hints**: ユーザーによる製品の特定の機能を識別するのに役立つアニメーション
- **Buttons**
 - **Next and back buttons** クイックスタートの各タスク内のステップおよびモジュールに移動するためのボタン
 - **Final screen buttons** クイックスタートを閉じたり、クイックスタート内の前のタスクに戻ったり、クイックスタートをすべて表示したりするためのボタン

クイックスタートの主なコンテンツエリアには、以下のセクションが含まれます。

- **Card copy**
- **はじめに**
- **タスクの手順**
- **Modals and in-app messaging**
- **Check your work module**

10.4. クイックスタートの継続

OpenShift Container Platform では、**ConsoleQuickStart** オブジェクトで定義されるクイックスタートのカスタムリソースが導入されています。Operator および管理者は、このリソースを使用してクイックスタートをクラスターに提供できます。

前提条件

- クラスター管理者の権限があること。

手順

1. 新規のクイックスタートを作成するには、以下を実行します。

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

2. 以下を実行します。

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

3. 本書で説明されているガイダンスを使用して、YAML ファイルを更新します。
4. 編集を保存します。

10.4.1. クイックスタート API ドキュメントの表示

手順

- クイックスタートの API ドキュメントを確認するには、以下を実行します。

```
$ oc explain consolequickstarts
```

oc explain の使用方法についての詳細は、**oc explain -h** を実行します。

10.4.2. クイックスタートの要素からクイックスタート CR へのマッピング

このセクションでは、クイックスタートのカスタムリソース (CR) の部分を、Web コンソール内のクイックスタートのこれらが表示される場所に視覚的にマッピングする方法を説明します。

10.4.2.1. conclusion 要素

YAML ファイルの conclusion 要素の表示

```
...
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 テキスト

Web コンソールでの conclusion 要素の表示

クイックスタートの最後のセクションに conclusion が表示されます。

Get started with Spring 10 minutes



- 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

Your Spring application is deployed and ready.

10.4.2.2. description 要素

YAML ファイルでの description 要素の表示

```
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
  ...
```

- 1 description テキスト

Web コンソールでの description 要素の表示

この description は、**Quick Starts** ページのクイックスタートの導入部分のタイルに表示されます。



Get started with Spring

🕒 10 minutes

Import a Spring Application from git, build, and deploy it onto OpenShift.

10.4.2.3. displayName 要素

YAML ファイルの displayName 要素の表示

```
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 1
  durationMinutes: 10
```

1 **displayName** テキスト。

Web コンソールでの displayName 要素の表示

表示名は、Quick Starts ページの導入部分のタイルに表示されます。



Get started with Spring

🕒 10 minutes

Import a Spring Application from git, build, and deploy it onto OpenShift.

10.4.2.4. durationMinutes 要素

YAML ファイルでの durationMinutes 要素の表示

```
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 ❶
```

- ❶ **durationMinutes** 値 (分単位)。この値は、クイックスタートの完了までにかかる時間を定義します。

Web コンソールでの durationMinutes 要素の表示

duration minutes 要素は、**Quick Starts** ページのクイックスタートの導入部分のタイルに表示されます。

YAML ファイルでの introduction 要素の表示

```
...
introduction: >- 1
  **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
...
```

1 introduction は、クイックスタートを紹介し、この中でタスクを一覧表示します。

Web コンソールでの introduction 要素の表示

クイックスタートカードをクリックすると、その中のサイドパネルスライドがクイックスタートを開始し、この中でタスクを一覧表示します。

Get started with Spring 10 minutes



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. カスタムアイコンのクイックスタートへの追加

デフォルトのアイコンがすべてのクイックスタートについて指定されます。独自のカスタムアイコンを指定することができます。

手順

1. カスタムアイコンとして使用する **.svg** ファイルを見つけます。
2. [オンラインツール](#)を使用して、**テキスト**を **base64** に変換 します。
3. YAML ファイルに **icon: >** を追加し、次の行に **data:image/svg+xml;base64** とそれに続く base64 変換からの出力が含まれます。以下に例を示します。

```
icon: >-  
  
data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdr  
cilHJvbGU9ImltZylgdmlld.
```

10.4.4. クイックスタートへのアクセス制限

すべてのユーザーがすべてのクイックスタートを利用できる訳ではありません。YAML ファイルの **accessReviewResources** セクションでは、クイックスタートへのアクセスを制限する機能を提供します。

ユーザーに **HelmChartRepository** リソースを作成する機能がある場合にのみクイックスタートにアクセスできるようにするには、以下の設定を使用します。

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

ユーザーに Operator グループおよびパッケージマニフェストを一覧表示し、Operator をインストールできる機能がある場合にのみクイックスタートにアクセスできるようにするには、以下の設定を使用します。

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

10.4.5. その他のクイックスタートのリンク

手順

- YAML ファイルの **nextQuickStart** セクションで、リンクするクイックスタートの **name** (**displayName** ではない) を指定します。以下に例を示します。

```
nextQuickStart:  
- add-healthchecks
```

10.4.6. クイックスタートでサポートされるタグ

これらのタグを使用して、クイックスタートコンテンツをマークダウンで記述します。マークダウンがHTMLに変換されます。

タグ	説明
'b',	太字テキストを定義します。
'img',	イメージを埋め込みます。
'i',	イタリックテキストを定義します。
'strike',	取り消し線 (strike-through) テキストを定義します。
's',	小さいテキストを定義します。
'del',	小さいテキストを定義します。
'em',	エミュレートしたテキストを定義します。
'strong',	重要なテキストを定義します。
'a',	アンカータグを定義します。
'p',	段落テキストを定義します。
'h1',	レベル1の見出しを定義します。
'h2',	レベル2の見出しを定義します。
'h3',	レベル3の見出しを定義します。
'h4',	レベル4の見出しを定義します。
'ul',	順序のないリストを定義します。
'ol',	順序付きのリストを定義します。
'li',	リスト項目を定義します。
'code',	テキストをコードとして定義します。

タグ	説明
'pre',	事前にフォーマットされたテキストのブロックを定義します。
'button',	テキストでボタンを定義します。

10.4.7. クイックスタートでのマークダウン参照の強調表示

ハイライトまたはヒントの機能により、クイックスタートに Web コンソールのコンポーネントを強調表示したり、アニメーションで表示できるリンクを追加することができます。

マークダウン構文には以下が含まれます。

- ブラケット付きリンクテキスト
- **highlight** のキーワードと、それに続くアニメーションで表示する要素の ID

10.4.7.1. パースペクティブスイッチャー

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

10.4.7.2. Administrator パースペクティブのナビゲーションリンク

```
[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 パースペクティブのナビゲーションリンク

```
[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. 一般的なナビゲーションリンク

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

10.4.7.5. マストヘッドリンク

```
[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. コードスニペットのマークダウン参照

CLI コードスニペットがクイックスタートに含まれる場合に、これを Web コンソールから実行できるようになりました。この機能を使用するには、まず Web Terminal Operator をインストールする必要があります。Web 端末で実行する Web 端末およびコードスニペットの各種アクションは、Web Terminal Operator をインストールしない場合は表示されません。または、Web Terminal Operator がインストールされているかどうかに関係なく、コードスニペットをクリップボードにコピーできます。

10.4.8.1. インラインコードスニペットの構文

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



注記

execute 構文が使用される場合、Web Terminal Operator がインストールされているかどうかに関係なく、**Copy to clipboard** アクションが表示されます。

10.4.8.2. 複数行コードスニペットの構文

```
...
multi line code block
```{{copy}}
...
multi line code block
```{{execute}}
```

10.5. クイックスタートのコンテンツガイドライン

10.5.1. Card copy

クイックスタートカードのタイトルと説明をカスタマイズできますが、ステータスをカスタマイズすることはできません。

- 説明を1または2文にまとめます。
- 動詞から始め、ユーザーの目的を伝えるものにします。正しい例:

```
Create a serverless application.
```

10.5.2. はじめに

クイックスタートカードをクリックすると、その中のサイドパネルスライドがクイックスタートを開始し、この中でタスクを一覧表示します。

- 導入部分のコンテンツを明確に、簡潔に、説明的に、また読みやすいものにします。
- クイックスタートの結果について示します。ユーザーは、クイックスタートを開始する前にその目的を理解している必要があります。
- ユーザーに (クイックスタートではなく) 実行するアクションを示します。

- **正しい例:**

In this quick start, you will deploy a sample application to {product-title}.

- **正しくない例:**

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

- 導入部分は、機能の複雑さに応じて最大 4 から 5 つの文章で設定される必要があります。導入部分が長いとユーザーを圧倒してしまう可能性があります。
- 導入部分の後にクイックスタートのタスクを一覧表示し、各タスクの一覧についてはそれぞれ動詞で始まります。タスクが追加または削除されるたびにコピーを更新する必要が生じるため、タスクの数は指定しないでください。

- **正しい例:**

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

- **正しくない例:**

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

10.5.3. タスクの手順

ユーザーが **Start** をクリックした後に、クイックスタートを完了するために実行する必要のある一覧のステップが表示されます。

タスクのステップを作成する場合は、以下の一般的なガイドラインに従います。

- ボタンとラベルには Click を使用します。チェックボックス、ラジオボタン、およびドロップダウンメニューには Select を使用します。
- Click on ではなく Click を使用します。

- **正しい例:**

Click OK.

- **正しくない例:**

Click on the OK button.

- ユーザーに対し、**Administrator** パースペクティブと **Developer** パースペクティブ間を移動する方法を示します。ユーザーがすでに適切なパースペクティブにいると思われる場合でも、ユーザーが適切なパースペクティブに確実に移動していることを確認できるように、ユーザーに対してパースペクティブへの移動方法を示します。

例:

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.

- Location, action の構造を使用します。ユーザーに対し、実行すべきアクションを示す前に移動する必要のある場所を示します。

- **正しい例:**

In the node.js deployment, hover over the icon.

- **正しくない例:**

Hover over the icon in the node.js deployment.

- 製品の用語については一貫して大文字表記を使用します。
- メニュータイプまたはリストをドロップダウンとして指定する必要がある場合は、ハイフンなしで dropdown と 1 単語で記述します。
- ユーザーアクションと製品機能に関する追加情報を明確に区別します。

- **ユーザーアクション:**

Change the time range of the dashboard by clicking the dropdown menu and selecting time range.

- **追加情報:**

To look at data in a specific time frame, you can change the time range of the dashboard.

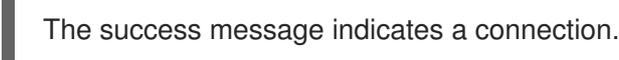
- 右上隅でアイコンをクリックなどの指示文は使用しないようにしてください。指示文は UI レイアウトが変更されるたびに古くなります。また、デスクトップユーザー向けの指示は、異なるサイズの画面を使用するユーザーには適切ではない場合があります。代わりに、名前を使用して内容を特定できるようにします。

- **正しい例:**

In the navigation menu, click Settings.

- **正しくない例:**

In the left-hand menu, click Settings.

- "Click the gray circle (灰色の円をクリック)" など、色のみで項目を特定することはしないでください。色の識別子は、視力制限のあるユーザー、とくに色覚異常のユーザーの役に立ちません。代わりに、ボタンコピーのような名前またはコピーを使用して項目を特定します。
 - **正しい例:**

 - **正しくない例:**

- 二人称を使用を統一します。
 - **正しい例:**

 - **正しくない例:**


10.5.4. 作業モジュールの確認

- ユーザーがステップを完了すると、**Check your work** モジュールが表示されます。このモジュールは、ユーザーに対してステップの結果についての質問への yes または no の回答を求めるプロンプトを出し、ユーザーはここで作業を確認することができます。このモジュールでは、1つの yes または no の回答を求める質問のみ作成する必要があります。
 - ユーザーが **Yes** と回答すると、チェックマークが表示されます。
 - ユーザーが **No** と回答すると、必要に応じて関連するドキュメントへのリンクと共にエラーメッセージが表示されます。その後、ユーザーは戻ってやり直すことができます。

10.5.5. UI 要素のフォーマット

以下のガイドラインを使用して UI 要素をフォーマットします。

- ボタン、ドロップダウン、タブ、フィールド、その他の UI コントロールのコピー: UI に表示されるようにコピーを作成し、これを太字にします。
- ページ、ウィンドウ、およびパネル名を含むその他のすべての UI 要素: UI に表示されるようにコピーを作成し、これを太字にします。
- コードまたはユーザーが入力するテキスト: 等幅フォントを使用します。
- ヒント: ナビゲーションまたはマストヘッド要素へのヒントが含まれる場合は、リンクのようにテキストのスタイルを変更します。
- CLI コマンド: 等幅フォントを使用します。
- 実行中のテキストで、コマンドに太字の等幅フォントを使用します。
- パラメーターまたはオプションが可変値である場合、イタリック体の等幅フォントを使用します。

- パラメーターに太字の等幅フォントを使用し、オプションに等幅フォントを使用します。

10.6. 関連情報

- 音声とトーンの要件については、[PatternFly のブランド音声およびトーンのガイドライン](#) について参照してください。
- 他の UX コンテンツのガイダンスは、[PatternFly の UX の作成ガイド \(writing style guide\)](#) のすべての分野を参照してください。