



Red Hat

OpenShift Container Platform 4.19

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 4.19 Web コンソール

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

Legal Notice

Copyright © 2025 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

このドキュメントでは、OpenShift Container Platform Web コンソールにアクセスしてカスタマイズする手順を説明します。

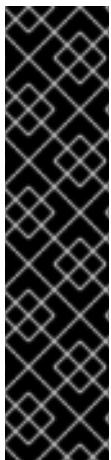
Table of Contents

第1章 WEB コンソールの概要	4
1.1. WEB コンソールの管理者ロール	4
1.2. WEB コンソールの開発者ロール	5
1.3. WEB コンソールで DEVELOPER パースペクティブを有効にする	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 OPENSHIFT CONTAINER PLATFORM DASHBOARD を使用したクラスター情報の取得	8
3.1. OPENSHIFT CONTAINER PLATFORM ダッシュボードページについて	8
3.2. リソースおよびプロジェクトの制限とクォータの認識	9
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 OPENSHIFT 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 OPENSHIFT 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	19
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	20
6.8. YAML サンプルの KUBERNETES リソースへの追加	20
6.9. ユーザーパースペクティブのカスタマイズ	21
6.10. 開発者カタログとサブカタログのカスタマイズ	24
第7章 動的プラグイン	28
7.1. 動的プラグインの概要	28
7.2. 動的プラグインを使い始める	31
7.3. クラスターへのプラグインのデプロイ	32
7.4. コンテンツセキュリティーポリシー (CSP)	36
7.5. 動的プラグインの例	38
7.6. 動的プラグイン参照	40
第8章 WEB 端末	118
8.1. WEB 端末のインストール	118
8.2. WEB 端末の設定	119
8.3. WEB 端末の使用	121
8.4. WEB 端末のトラブルシューティング	122
8.5. WEB 端末のアンインストール	123
第9章 OPENSHIFT CONTAINER PLATFORM の WEB コンソールの無効化	127
9.1. 前提条件	127
9.2. WEB コンソールの無効化	127
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	128
10.1. クイックスタートについて	128

10.2. クイックスタートのユーザーウォークフロー	128
10.3. クイックスタートのコンポーネント	129
10.4. クイックスタートの継続	129
10.5. クイックスタートのコンテンツガイドライン	141
10.6. 関連情報	145
第11章 WEB コンソールのオプション機能と製品	146
11.1. OPERATOR を使用して OPENSHIFT CONTAINER PLATFORM WEB コンソールを拡張する	146
11.2. WEB コンソールでの RED HAT OPENSHIFT LIGHTSPEED	146
11.3. WEB コンソールの RED HAT OPENSHIFT PIPELINE	146
11.4. WEB コンソールの RED HAT OPENSHIFT SERVERLESS	146
11.5. OPENSHIFT CONTAINER PLATFORM WEB コンソールの RED HAT DEVELOPER HUB	147

第1章 WEB コンソールの概要

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、運営、トラブルシューティングのタスクを実行するための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。



重要

OpenShift Container Platform 4.19 以降、Web コンソールのパースペクティブが統合されました。**Developer** パースペクティブは、デフォルトでは有効になっていません。

すべてのユーザーは、OpenShift Container Platform Web コンソールのすべての機能を操作できます。ただし、クラスターの所有者でない場合は、特定の機能にアクセスする権限をクラスターの所有者に要求する必要がある場合があります。

引き続き **Developer** パースペクティブを有効にできます。Web コンソールの [Getting Started](#) ペインでは、コンソールツールの実行、クラスター設定に関する情報の検索、**Developer** パースペクティブを有効にするためのクイックスタートの表示、リンク先を表示して新機能の確認などを行えます。

ユーザータスクを使用して Web コンソール内でガイド付きの手順を提供する、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。これらは、アプリケーション、Operator、またはその他のオファリングの概要を理解するのに役立ちます。

1.1. WEB コンソールの管理者ロール

クラスター管理者ロールを使用すると、クラスターのインベントリー、容量、全体的および詳細な使用状況、そして重要なイベントの流れを表示できます。これらはすべて、計画立案やトラブルシューティングのタスク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はどちらも、Web コンソールのすべての機能を使用できます。

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。

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。
- OperatorHub を使用して Operator をインストールおよび管理します。
- 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。
- クラスターに関するロギング、メトリック、および高ステータスの情報を表示し、管理する。
- アプリケーション、コンポーネント、サービスと視覚的に対話します。

1.2. WEB コンソールの開発者ロール

Web コンソールの開発者ロールには、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のいくつかの組み込みの方法が提供されています。開発者ロールを使用すると、次のことが可能になります。

- 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ックを検査して、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。メトリックにより、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。

開発者は、次のようなユースケースに固有のワークフローにアクセスできます。

- 既存のコードベース、イメージ、およびコンテナー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。
- サーバーレス機能 (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構成の表示](#) を参照してください。

1.3. WEB コンソールで DEVELOPER パースペクティブを有効にする

OpenShift Container Platform 4.19 以降、Web コンソールのパースペクティブが統合されました。デフォルトでは Developer パースペクティブは存在しなくなりましたが、クラスター管理者は開発者が使用できるように Developer パースペクティブを有効化できます。

Developer パースペクティブを有効化するには、次の手順を実行します。

前提条件

- **cluster-admin** 権限を持つユーザーとして Web コンソールにアクセスできる。

手順

1. Administration → Cluster Settings をクリックして、Cluster Settings ページに移動します。
2. Cluster Settings ページで、Configuration タブを選択します。
3. 検索の際に **console** と入力して Console Operator リソースを見つけ、operator.openshift.io を選択します。
4. Cluster Details ページで、Actions メニューをクリックし、Customize を選択します。
5. General タブで、Perspectives セクションを見つけます。必要に応じて、Developer パースペクティブを有効化または無効化できます。変更は自動的に適用されます。
6. オプション: 次のコマンドを実行すると、CLI で Developer パースペクティブを有効化できます。

```
$ oc patch console.operator.openshift.io/cluster --type='merge' -p '{"spec":{"customization":{"perspectives":[{"id":"dev","visibility":{"state":"Enabled"}}]}}}'
```



注記

コンソール Pod が再起動するため、変更が Web コンソールに反映されるまでに少し時間がかかります。

関連情報

- [クラスター管理者について](#)
- [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 コンソールについて](#)
- [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。Web コンソールを使用して、プロジェクトの内容を視覚化、参照、管理できます。

2.1. 前提条件

- サポートされている Web ブラウザー (Edge、Chrome、Safari、または Mozilla Firefox) のいずれかを使用している。IE 11以前はサポートされていません。
- [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、コントロールプレーンノード上で 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-install create cluster コマンドを使用して OpenShift Container Platform をインストールした後、インストールプログラムの CLI 出力で、インストールされたクラスターの Web コンソールの URL とログイン認証情報を確認できます。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。



重要

dir パラメーターは、マニフェストファイル、ISO イメージ、および **auth** ディレクトリーや **assets** ディレクトリを保存する **assets** ディレクトリを指定します。**auth** ディレクトリには、**kubeadmin-password** および **kubeconfig** ファイルが保存されます。**kubeadmin** ユーザーとして、設定 **export KUBECONFIG=<install_directory>/auth/kubeconfig** で **kubeconfig** ファイルを使用して、クラスターにアクセスできます。**kubeconfig** は生成された ISO イメージに固有であるため、**kubeconfig** が設定されていて、**oc** コマンドが失敗した場合は、システムが生成された ISO イメージで起動しなかった可能性があります。デバッグを実行するには、ブートストラッププロセス中に、**kubeadmin-password** ファイルの内容を使用して、**core** ユーザーとしてコンソールにログインできます。

関連情報

- [Web コンソールを使用した機能セットの有効化](#)

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

OpenShift Container Platform の Web コンソールは、クラスターに関する概要情報を取得します。

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

OpenShift Container Platform Web コンソールから **Home** → **Overview** に移動して、クラスターに関する概要情報を取得する 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。

OpenShift Container Platform ダッシュボードは以下のカードで構成されます。

- **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**、**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、リソースの数と関連するステータスの詳細を示します。これは、問題の解決に介入が必要な場合に役立ちます。以下に関する情報が含まれます。
 - ノード数
 - Pod 数
 - 永続ストレージボリューム要求
 - 状態別にリスト表示されたクラスター内のベアメタルホスト (metal3 環境でのみ利用可能)
- **Status** 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。
- **Cluster Utilization** には、指定期間におけるさまざまなリソースの容量が表示されます。これは、リソース消費量が多い場合に、管理者がその規模と頻度を把握するのに役立ちます。次の情報が表示されます。
 - CPU 時間
 - メモリー割り当て
 - 消費されたストレージ
 - 消費されたネットワークリソース
 - Pod 数
- **Activity** には、Pod の作成や別のホストへの仮想マシンの移行など、クラスター内の最近のアクティビティーに関するメッセージがリスト表示されます。

3.2. リソースおよびプロジェクトの制限とクオータの認識

Web コンソールの **Developer** パースペクティブの **Topology** ビューで、利用可能なリソースのグラフィカル表示を使用できます。

リソースの制限やクオータに到達したことを示すメッセージがリソースにある場合は、リソース名の周囲に黄色の境界線が表示されます。メッセージを表示するには、リソースをクリックしてサイドパネルを開きます。Topology ビューがズームアウトされている場合、黄色の点はメッセージがあることを示します。

View Shortcuts メニューから **List View** を使用すると、リソースのリストが表示されます。 **Alerts** 列は、メッセージがあるかどうかを示します。

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(グラフまたはリスト)、編集メディア(フォームまたは YAML)、言語設定、およびリソースタイプを設定できます。

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. **Theme** フィールドで、作業するテーマを設定できます。ログインするたびに、選択したテーマがコンソールのデフォルトとして設定されます。
 - b. **Project** フィールドで、作業するプロジェクトを選択します。ログインするたびに、そのプロジェクトがコンソールのデフォルトとして設定されます。
 - 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、リスト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、**Language** 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。
5. **Notifications** セクションでは、**Overview** ページまたは通知ドロワーで、特定のプロジェクトに対してユーザーが作成した通知の表示を切り替えることができます。
6. **Applications** セクションで、以下を実行します。
 - a. デフォルトの **リソースタイプ** を表示できます。たとえば、OpenShift Serverless Operator がインストールされている場合、デフォルトのリソースタイプは **Serverless Deployment** です。それ以外の場合、デフォルトのリソースタイプは **Deployment** です。
 - b. **リソースタイプ** フィールドから、別のリソースタイプをデフォルトのリソースタイプとして選択できます。

第5章 OPENSHIFT 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、クイックスタートチュートリアルを無効にしたりできます。

5.1. 前提条件

- OpenShift Container Platform クラスターをデプロイしている。

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
```

- 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ー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。
- 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

- `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: 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: 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つのクリックスタートを有効または無効にするには、該当するクリ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リックスタートを適切なリストに移動します。
 - 複数のクリ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リ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リックスタートを適切なリストに移動します。
 - すべてのクリ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、すべてのクリックスタートを適切なリストに移動します。

第6章 OPENSHIFT 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

- 管理者権限を持っている。
- 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。ConfigMap オブジェクトサイズの制約により、イメージサイズは1MB を超えてはなりません。

手順

- 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> ... ①
```

- ① 有効な base64 でエンコードされたロゴを指定します。

- 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。

- 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

- 管理者権限を持っている。

手順

- Administration → Custom Resource Definitions から、ConsoleLink をクリックします。
- Instances タブを選択します。
- 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 ①
  text: Link 1
```

① 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

- クラスター **Ingress** 設定を編集します。

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

- 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。

- 変更を適用するためにファイルを保存します。



注記

アプリケーション Ingress ロードバランサーを指すカスタムコンソールルートの DNS レコードを追加します。

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

- クラスター **Ingress** 設定を編集します。

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

- 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。

- 変更を適用するためにファイルを保存します。



注記

アプリケーション Ingress ロードバランサーを指すカスタムダウンロードルートの DNS レコードを追加します。

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

- 管理者権限を持っている。

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。

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
# ...
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、特定の方法で 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

- 管理者権限を持っている。

手順

- Administration → Custom Resource Definitions から、ConsoleExternalLogLink をクリックします。
- Instances タブを選択します。
- 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. カスタム通知バーの作成

前提条件

- 管理者権限を持っている。

手順

- Administration → Custom Resource Definitions から、ConsoleNotification をクリックします。
- Instances タブを選択します。
- 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

- 管理者権限を持っている。

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) のリストから **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
spec:
  description: |
    This is an example of download links
  displayName: example
  links:
    - href: 'https://www.example.com/public/example.tar'
      text: example for linux
    - href: 'https://www.example.com/public/example.mac.zip'
      text: example for mac
    - href: 'https://www.example.com/public/example.win.zip'
      text: example 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

- クラスター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。

6.9. ユーザーパースペクティブのカスタマイズ

OpenShift Container Platform Web コンソールは、デフォルトで **Administrator** と **Developer** の 2 つのパースペクティブを提供します。インストールされているコンソールプラグインによっては、より多くのパースペクティブを使用できる場合があります。クラスター管理者は、すべてのユーザーまたは特定のユーザー ロールのパースペクティブを表示または非表示にすることができます。パースペクティブをカスタマイズすると、ユーザーは自分のロールとタスクに適用できるパースペクティブのみを表示できるようになります。たとえば、権限のないユーザーがクラスターリソース、ユーザー、およびプロジェクトを管理できないように、Administrator パースペクティブを非表示にすることができます。同様に、開発者ロールを持つユーザーに Developer パースペクティブを表示して、アプリケーションを作成、デプロイ、および監視できるようにすることができます。

ロールベースのアクセス制御 (RBAC) に基づいて、ユーザーのパースペクティブの表示をカスタマイズすることもできます。たとえば、特定の権限を必要とする監視目的でパースペクティブをカスタマイズする場合、パースペクティブが必要な権限を持つユーザーにのみ表示されるように定義できます。

各パースペクティブには、YAML ビューで編集できる次の必須パラメーターが含まれています。

- **id**: 表示または非表示にするパースペクティブの ID を定義します
- **visibility**: パースペクティブの状態と、必要に応じてアクセスレビュー・チェックを定義します。
- **state**: パースペクティブが有効か、無効か、アクセスレビュー・チェックが必要かを定義します



注記

デフォルトでは、すべてのパースペクティブが有効になっています。ユーザーパースペクティブをカスタマイズすると、その変更はクラスター全体に適用されます。

6.9.1. YAML ビューを使用したパースペクティブのカスタマイズ

前提条件

- 管理者権限を持っている。

手順

1. Administrator パースペクティブで、Administration → Cluster Settings に移動します。
2. Configuration タブを選択し、Console (operator.openshift.io) リソースをクリックします。
3. YAML タブをクリックして、カスタマイズを行います。
 - a. パースペクティブを有効または無効にするには、Add user perspectives のスニペットを挿入し、必要に応じて YAML コードを編集します。

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
  customization:
    perspectives:
      - id: admin
        visibility:
          state: Enabled
      - id: dev
        visibility:
          state: Enabled
```

- b. RBAC 権限に基づいてパースペクティブを非表示にするには、ユーザーパースペクティブを非表示にするためのスニペットを挿入し、必要に応じて YAML コードを編集します。

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
  customization:
    perspectives:
      - id: admin
        requiresAccessReview:
          - group: rbac.authorization.k8s.io
```

```

resource: clusterroles
verb: list
- id: dev
  state: Enabled

```

c. ニーズに基づいてパースペクティブをカスタマイズするには、独自の YAML スニペットを作成します。

```

apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
  customization:
    perspectives:
      - id: admin
        visibility:
          state: AccessReview
        accessReview:
          missing:
            - resource: deployment
              verb: list
          required:
            - resource: namespaces
              verb: list
      - id: dev
        visibility:
          state: Enabled

```

4. **Save** をクリックします。

6.9.2. フォームビューを使用したパースペクティブのカスタマイズ

前提条件

- 管理者権限を持っている。

手順

- Administrator パースペクティブで、Administration → Cluster Settings に移動します。
- Configuration タブを選択し、Console (operator.openshift.io) リソースをクリックします。
- ページの右側にある Actions → Customize をクリックします。
- General 設定で、ドロップダウンリストから次のいずれかのオプションを選択して、パースペクティブをカスタマイズします。
 - Enabled:** すべてのユーザーに対してパースペクティブを有効にします
 - Only visible for privileged users:** すべての namespace を一覧表示できるユーザーのパースペクティブを有効にします。
 - Only visible for unprivileged users:** すべての namespace を一覧表示できないユーザーのパースペクティブを有効にします。

- **Disabled:** すべてのユーザーに対してパースペクティブを無効にします
変更が保存されたことを確認する通知が開きます。

The screenshot shows the 'Perspectives' configuration page. On the left, there is a sidebar with 'General', 'Projects', and 'Developer' options. The 'Developer' option is selected. The main area has a table with two rows. The first row has 'Administrator' in the first column and 'Enabled' in the second column. The second row has 'Developer' in the first column and 'Disabled' in the second column. Below the table, a green box contains the text 'Saved.' and a note: 'This config update requires a console rollout, this can take up to a minute and require a browser refresh.'



注記

ユーザーパースペクティブをカスタマイズすると、変更は自動的に保存され、ブラウザーの更新後に有効になります。

6.10. 開発者カタログとサブカタログのカスタマイズ

クラスター管理者は、Developer カタログまたはそのサブカタログを編成および管理できます。サブカタログタイプを有効または無効にするか、開発者カタログ全体を無効にすることができます。

developerCatalog.types オブジェクトには、YAML ビューで使用するためにスニペットで定義する必要がある次のパラメーターが含まれています。

- **state:** 開発者カタログタイプのリストを有効にするか無効にするかを定義します。
- **enabled:** ユーザーに表示される開発者カタログタイプ(サブカタログ)のリストを定義します。
- **disabled:** ユーザーに表示されない開発者カタログタイプ(サブカタログ)のリストを定義します。

YAML ビューまたはフォームビューを使用して、次の開発者カタログタイプ(サブカタログ)を有効または無効にすることができます。

- **Builder Images**
- **テンプレート**
- **Devfiles**
- **Samples**
- **Helm Charts**
- **Event Sources**
- **Event Sinks**
- **Operator Backed**

6.10.1. YAML ビューを使用した開発者カタログまたはそのサブカタログのカスタマイズ

YAML ビューで YAML コンテンツを編集することにより、開発者カタログをカスタマイズできます。

前提条件

- クラスター管理者権限を持つ OpenShift Web コンソールセッション。

手順

1. Web コンソールの **Administrator** パースペクティブで、Administration → Cluster Settings に移動します。
2. Configuration タブを選択し、Console (operator.openshift.io) リソースをクリックして、Details ページを表示します。
3. YAML タブをクリックしてエディターを開き、必要に応じて YAML コンテンツを編集します。たとえば、開発者カタログタイプを無効にするには、無効な開発者カタログリソースのリストを定義する次のスニペットを挿入します。

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
...
spec:
  customization:
    developerCatalog:
      categories:
        types:
          state: Disabled
          disabled:
            - BuilderImage
            - Devfile
            - HelmChart
...
...
```

4. Save をクリックします。



注記

デフォルトでは、Web コンソールの管理者ビューで開発者カタログタイプが有効になっています。

6.10.2. フォームビューを使用した開発者カタログまたはそのサブカタログのカスタマイズ

Web コンソールのフォームビューを使用して、開発者カタログをカスタマイズできます。

前提条件

- クラスター管理者権限を持つ OpenShift Web コンソールセッション。
- Developer パースペクティブが有効になっている。

手順

1. Administrator パースペクティブで、Administration → Cluster Settings に移動します。
2. Configuration タブを選択し、Console (operator.openshift.io) リソースをクリックします。
3. Actions → Customize をクリックします。
4. Pre-pinned navigation items、Add page、および Developer Catalog セクションの項目を有効または無効にします。

検証

開発者カタログをカスタマイズすると、変更内容がシステムに自動的に保存され、更新後にブラウザーで有効になります。

 **Saved.**

This config update requires a console rollout, this can take up to a minute and require a browser refresh.



注記

管理者は、すべてのユーザーに対してデフォルトで表示されるナビゲーション項目を定義できます。ナビゲーション項目を並べ替えることもできます。

ヒント

同様の手順を使用して、クイックスタート、クラスターロール、アクションなどの Web UI 項目をカスタマイズできます。

6.10.2.1. YAML ファイルの変更例

開発者カタログをカスタマイズするために、YAML エディターに次のスニペットを動的に追加できます。

次のスニペットを使用して、状態 タイプを Enabled に設定してすべてのサブカタログを表示します。

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
...
spec:
  customization:
    developerCatalog:
      categories:
        types:
          state: Enabled
```

次のスニペットを使用して、状態 タイプを Disabled に設定してすべてのサブカタログを無効にします。

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

```

```
spec:  
  customization:  
    developerCatalog:  
      categories:  
      types:  
        state: Disabled
```

クラスター管理者が、Web コンソールで有効になっているサブカタログのリストを定義する場合は、次のスニペットを使用します。

```
apiVersion: operator.openshift.io/v1  
kind: Console  
metadata:  
  name: cluster  
...  
spec:  
  customization:  
    developerCatalog:  
      categories:  
      types:  
        state: Enabled  
        enabled:  
        - BuilderImage  
        - Devfile  
        - HelmChart  
        - ...
```

第7章 動的プラグイン

7.1. 動的プラグインの概要

7.1.1. 動的プラグインについて

動的プラグインは実行時にリモートのソースからロードされ、解釈されます。動的プラグインをコンソールに提供して公開する1つの方法は、OLM Operatorを使用することです。このOperatorは、プラグインをホストするHTTPサーバーを備えたプラットフォーム上にデプロイメントを作成し、Kubernetesサービスを使用してそれを公開します。

動的プラグインを使用すると、実行時にカスタムページやその他の拡張機能をコンソールユーザー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にプラグインを登録し、クラスター管理者はコンソールのOperator設定でプラグインを有効にします。

7.1.2. 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。
- 管理者と開発者を超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。

7.1.3. 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、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。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要素セレクターなどのセレクターを回避します。
- プラグインWebサーバーが提供するすべてのアセットの**Content-Type** レスポンスヘッダーを使用して、有効な JavaScript Multipurpose Internet Mail Extension (MIME) タイプを指定しま

す。各プラグインデプロイメントには、そのプラグインの生成済みアセットをホストする Web サーバーが含まれている必要があります。

- Webpack バージョン 5 以降を使用して、Webpack でプラグインをビルドします。
- 競合を避けるために、CSS クラス名の前にプラグイン名を付けます。例: **my-plugin_heading** および **my-plugin_icon**
- 他のコンソールページと一貫性のある外観、操作性、動作を維持します。
- 要素セレクターなど、プラグインコンポーネント外部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。
- プラグイン Web サーバーによって提供されるすべてのアセットに対して、**Content-Type** レスポンスヘッダーを使用して有効な JavaScript Multipurpose Internet Mail Extension (MIME) タイプを指定する必要があります。各プラグインデプロイメントには、そのプラグインの生成済みアセットをホストする Web サーバーが含まれている必要があります。

7.1.4. PatternFly 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。

- [PatternFly コンポーネント](#)と 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。PatternFly コンポーネントと変数を使用すると、今後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。
 - OpenShift Container Platform バージョン 4.14 以前を使用している場合は、Patternfly 4.x を使用します。
 - OpenShift Container Platform 4.15 以降を使用している場合は、Patternfly 5.x を使用します。
- [PatternFly's accessibility fundamentals](#) 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 と競合し、コンソールの他の部分とマッチしない可能性があります。プラグインには、基本の PatternFly スタイルに加え、評価対象のユーザーインターフェイスに固有のスタイルのみを含めます。@patternfly/react-styles/*/.css などのスタイルや @patternfly/patternfly パッケージからのスタイルは、プラグインにインポートしないでください。
- コンソールアプリケーションは、サポートされているすべての PatternFly バージョンの基本スタイルをロードします。

7.1.4.1. react-i18next でメッセージを翻訳する

この [プラグインテンプレート](#) は、react-i18next を使用してメッセージを翻訳する方法を示しています。

前提条件

- プラグインテンプレートをローカルに複製する必要があります。
- オプション: プラグインをローカルでテストするには、コンテナー内で OpenShift Container Platform Web コンソールを実行します。Docker または Podman 3.2.0 以降を使用できます。

手順

- 名前の競合を避けるために、名前の前に **plugin__** を付けます。このプラグインテンプレートは、デフォルトで **plugin__console-plugin-template** namespace を使用します。プラグインの名前(例: **plugin__my-plugin**)を変更する場合は更新する必要があります。たとえば、**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>;
};
```



重要

i18n namespace を **ConsolePlugin** リソースの名前と一致させる必要があります。

- 必要な動作に応じて **spec.i18n.loadType** フィールドを設定します。

例7.1 **plugin__console-demo-plugin**

```
spec:
  backend:
    service:
      basePath: /
      name: console-demo-plugin
      namespace: console-demo-plugin
      port: 9001
      type: Service
      displayName: OpenShift Console Demo Plugin
  i18n:
    loadType: Preload ①
```

- ロード中、動的プラグインの後に、**i18n** namespace からプラグインのすべてのローカリゼーションリソースを読み込みます。

- console-extensions.json** 内のラベルに **%plugin__console-plugin-template~My Label%** という形式を使用します。コンソールは、値を **plugin__console-plugin-template** namespace の現在の言語のメッセージに置き換えます。以下に例を示します。

```
{
  "type": "console.navigation/section",
  "properties": {
    "id": "admin-demo-section",
    "perspective": "admin",
    "name": "%plugin__console-plugin-template~Plugin Template%"
  }
}
```

- i18next-parser** の TypeScript ファイルにコメントを追加して、**console-extensions.json** からのメッセージをメッセージカタログに追加します。以下に例を示します。

```
// t('plugin__console-demo-plugin~Demo Plugin')
```

- メッセージを追加または変更するときにプラグインテンプレートの **locales** フォルダー内の JSON ファイルを更新するために、次のコマンドを実行します。

```
$ yarn i18n
```

7.2. 動的プラグインを使い始める

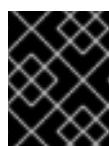
動的プラグインの使用を開始するには、新しい OpenShift Container Platform 動的プラグインを作成するように環境をセットアップする必要があります。新しいプラグインを作成する方法の例は、[Pod ページへのタブの追加](#) を参照してください。

7.2.1. 動的プラグインの開発

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 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ーで実行されます。

前提条件

- 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る [console-plugin-template](#) リポジトリを複製しておく。



重要

Red Hat はカスタムプラグインコードをサポートしていません。プラグインで利用できるのは、[共同コミュニティーのサポート](#) のみです。

- OpenShift Container Platform クラスターが実行している必要があります。
- OpenShift CLI (**oc**) がインストールされている。
- yarn** がインストールされている。
- Docker v3.2.0 以降または Podman v3.2.0 以降がインストール済みで実行中である。

手順

- ターミナルウィンドウを 2 つ開きます。
- 一方のターミナルウィンドウで、次のコマンドを実行し、yarn を使用してプラグインの依存関係をインストールします。

```
$ yarn install
```

- インストール後、以下のコマンドを実行して yarn を起動します。

```
$ yarn run start
```

- 別のターミナルウィンドウで、CLI 経由で OpenShift Container Platform Web コンソールにログインします。

```
$ oc login
```

- 以下のコマンドを実行して、ログインしたクラスターに接続されたコンテナーで OpenShift Container Platform Web コンソールを実行します。

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



注記

yarn run start-console コマンドは、**amd64** イメージを実行します。Apple Silicon および Podman で実行すると、失敗する可能性があります。これは、以下のコマンドを実行して、**qemu-user-static** を使用すると回避できます。

```
$ podman machine ssh  
$ sudo -i  
$ rpm-ostree install qemu-user-static  
$ systemctl reboot
```

検証

- [localhost:9000](#) にアクセスして、実行中のプラグインを表示します。**window.SERVER_FLAGS.consolePlugins** の値を検査し、ランタイム時にロードされるプラグインの一覧を表示します。

7.3. クラスターへのプラグインのデプロイ

プラグインを OpenShift Container Platform クラスターにデプロイできます。

7.3.1. Docker を使用したイメージのビルト

クラスターにプラグインをデプロイするには、まずイメージをビルトし、そ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

- 以下のコマンドでイメージをビルトします。

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

- 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 -p 9001:80 quay.io/my-repository/my-plugin:latest
```

- 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
```

7.3.2. クラスターへのプラグインのデプロイ

変更を加えたイメージをレジストリーにプッシュした後、Helm チャートを使用してプラグインをクラスターにデプロイできます。

前提条件

- 以前にプッシュしたプラグインを含むイメージの場所が用意されている。



注記

プラグインのニーズに応じて追加のパラメーターを指定できます。[values.yaml](#) ファイルに、サポートされているパラメーターがすべて含まれています。

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、以下のようになります。

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。



注記

OpenShift Container Platform 4.10 以降にデプロイする場合は、**--set plugin.securityContext.enabled=false** パラメーターを追加して、Pod セキュリティに関連する設定を除外することを推奨します。

- 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

- 有効なプラグインのリストを表示するには、Administration → Cluster Settings → Configuration → **Console operator.openshift.io** → Console plugins に移動するか、Overview ページにアクセスします。



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生じる可能性があります。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。

7.3.3. プラグインサービスプロキシー

プラグインからクラスター内のサービスに HTTP リクエストを送信する必要がある場合は、**spec.proxy** 配列フィールドを使用して、**ConsolePlugin** リソースでサービスプロキシーを宣言できます。コンソールバックエンドは、プラグインとサービス間の通信をプロキシーするため `/api/proxy/plugin/<plugin-name>/<proxy-alias>/<request-path>?<optional-query-parameters>` エンドポイントを公開します。プロキシーされたリクエストは、デフォルトで サービス CA バンドルを使用します。このサービスは HTTPS を使用する必要があります。



注記

プラグインは、JavaScript コードからリクエストを行うために **consolefetch** API を使用する必要があります。そうしないと、一部のリクエストが失敗する可能性があります。 詳細は、「動的プラグイン API」を参照してください。

エントリーごとに、**endpoint** and **alias** フィールドでプロキシーのエンドポイントとエイリアスを指定する必要があります。Service プロキシータイプの場合、エンドポイント **type** フィールドを **Service** に設定し、**service** に **name**、**namespace**、および **port** フィールドの値を含める必要があります。たとえば、`/api/proxy/plugin/helm/helm-charts/releases?limit=10` は、10 個の Helm リリースをリストする **helm-charts** サービスが含まれる **helm** プラグインからのプロキシー要求パスです。

サービスプロキシーの例

```
apiVersion: console.openshift.io/v1
kind: ConsolePlugin
metadata:
  name:<plugin-name>
spec:
  proxy:
    - alias: helm-charts ①
      authorization: UserToken ②
      caCertificate: '-----BEGIN CERTIFICATE-----\nMIID....'en ③
      endpoint: ④
      service:
        name: <service-name>
        namespace: <service-namespace>
        port: <service-port>
        type: Service
```

① プロキシーのエイリアス。

② ログインしたユーザーの OpenShift Container Platform アクセストークンをサービスプロキシー要求に含める必要がある場合は、authorization フィールドを **UserToken** に設定する必要があります。



注記

ログインしたユーザーの OpenShift Container Platform アクセストークンがサービスプロキシー要求に含まれていない場合は、authorization フィールドを **None** に設定してください。

- ③ サービスがカスタムサービス CA を使用する場合、**caCertificate** フィールドに証明書バンドルが含まれている必要があります。
- ④ プロキシーのエンドポイント。

関連情報

- [サービス CA 証明書](#)
- [サービス提供証明書のシークレットによるサービストラフィックのセキュリティー保護](#)
- [動的プラグイン API](#)

7.3.4. ブラウザーでのプラグインの無効化

コンソールユーザーは、**disable-plugins** クエリーパラメーターを使用して、通常ランタイム時にロードされる特定またはすべての動的プラグインを無効にすることができます。

手順

- 特定のプラグインを無効にするには、プラグイン名のコンマ区切りリストから無効にするプラグインを削除します。
- すべてのプラグインを無効にするには、**disable-plugins** クエリーパラメーターを空の文字列のままにします。



注記

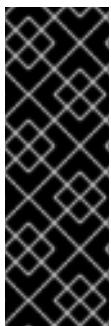
クラスター管理者は、Web コンソールの **Cluster Settings** ページでプラグインを無効にできます。

7.3.5. 関連情報

- [Helm について](#)

7.4. コンテンツセキュリティーポリシー (CSP)

ConsolePluginSpec ファイルの **contentSecurityPolicy** フィールドを使用して、動的プラグインのコンテンツセキュリティーポリシー (CSP) ディレクティブを指定できます。このフィールドは、スクリプト、スタイル、イメージ、フォントなどのコンテンツの取得を許可するソースを指定することにより、潜在的なセキュリティーリスクを軽減するのに役立ちます。外部ソースからリソースをロードする必要がある動的プラグインがある場合は、カスタム CSP ルールを定義すると、OpenShift Container Platform コンソールへのセキュアな統合を実現できます。



重要

現在、コンソールは **Content-Security-Policy-Report-Only** レスポンスヘッダーを使用しています。そのため、ブラウザーの Web コンソールで CSP 違反について警告されるだけであり、CSP ポリシーの適用は限られています。CSP 違反はブラウザーのコンソールに記録されますが、関連する CSP ディレクティブは適用されません。この機能は **feature-gate** の背後にありますため、手動で有効にする必要があります。

詳細は、[Web コンソールを使用した機能セットの有効化](#) を参照してください。

7.4.1. コンテンツセキュリティポリシー (CSP) の概要

コンテンツセキュリティポリシー (CSP) は、**Content-Security-Policy-Report-Only** レスポンスヘッダーでブラウザーに配信されます。ポリシーは、一連のディレクティブと値の形で指定します。ディレクティブタイプによって果たす目的が異なります。各ディレクティブには、許可されるソースを表す値のリストを含めることができます。

7.4.1.1. contentSecurityPolicy の主な機能

7.4.1.1.1. ディレクティブタイプ

サポートされているディレクティブタイプには、**DefaultSrc**、**ScriptSrc**、**StyleSrc**、**ImgSrc**、**FontSrc** があります。これらのディレクティブを使用すると、プラグインにさまざまな種類のコンテンツを読み込むための有効なソースを指定できます。ディレクティブタイプによって果たす目的が異なります。たとえば、**ScriptSrc** は有効な JavaScript ソースを定義し、**ImgSrc** はイメージを読み込む場所を制御します。

7.4.1.1.2. 値

各ディレクティブには、許可されるソースを表す値のリストを含めることができます。たとえば、**ScriptSrc** では複数の外部スクリプトを指定できます。これらの値は 1024 文字に制限されており、空白、コンマ、セミコロンを含めることはできません。また、一重引用符で囲まれた文字列とワイルドカード文字 (*) は使用できません。

7.4.1.1.3. ポリシーの統合

OpenShift Container Platform Web コンソールは、すべての有効な **ConsolePlugin** カスタムリソース (CR) の CSP ディレクティブを集約し、独自のデフォルトポリシーと結合します。結合されたポリシーは、**Content-Security-Policy-Report-Only** HTTP レスポンスヘッダーによって適用されます。

7.4.1.1.4. 検証ルール

- 各ディレクティブには最大 16 個の一意の値を設定できます。
- ディレクティブ全体の値の合計サイズが、8192 バイト (8 KB) を超えないようにする必要があります。
- 各値は一意である必要があります。また、引用符、スペース、コンマ、またはワイルドカード記号が使用されていないことを確認する追加の検証ルールが適用されます。

7.4.2. 関連情報

- [コンテンツセキュリティポリシー \(CSP\)](#)

7.5. 動的プラグインの例

例を実行する前に、[動的プラグイン開発](#) の手順に従って、プラグインが機能していることを確認してください。

7.5.1. Pod ページへのタブの追加

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。以下の手順では、サンプルとしてプラグインにタブを **Pod details** ページに追加します。



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ーで実行されます。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ための情報は、「[動的プラグインの開発](#)」を参照してください。

手順

- 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を含む [console-plugin-template](#) リポジトリにアクセスします。



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。

- 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。
- 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。
- 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。
- 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。

7.6. 動的プラグイン参照

プラグインのカスタマイズを可能にするエクステンションを追加できます。これらのエクステンションは、ランタイム時にコンソールにロードされます。

7.6.1. 動的プラグインエクステンションのタイプ

7.6.1.1. console.action/filter

ActionFilter を使用してアクションを絞り込むことができます。

名前	値のタイプ	任意	説明
contextId	string	いいえ	コンテキスト ID は、提供したアクションのスコープをアプリケーションの特定のエリアに限定するのに役立ちますたとえば、トポロジー および helm などがあります。
filter	CodeRef<(スコープ: any、 action: Action) ⇒ boolean>	いいえ	<p>一部の条件に基づいてアクションをフィルターする関数。</p> <p>scope: アクションを指定するスコープ。 Horizontal Pod Autoscaler (HPA) のデプロイメントから</p> <p>ModifyCount アクションを削除する必要がある場合には、フックが必要になることがあります。</p>

7.6.1.2. console.action/group

ActionGroup は、サブメニューに指定可能なアクショングループを提供します。

名前	値のタイプ	任意	説明
id	string	いいえ	アクションの選択を識別するための ID。

名前	値のタイプ	任意	説明
<code>label</code>	<code>string</code>	はい	UI に表示されるラベル。サブメニューに必要です。
<code>submenu</code>	<code>boolean</code>	はい	このグループをサブメニューとして表示するかどうか。
<code>insertBefore</code>	<code>string string[]</code>	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
<code>insertAfter</code>	<code>string string[]</code>	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 <code>insertBefore</code> 値が優先されます。

7.6.1.3. `console.action/provider`

`ActionProvider` は、特定のコンテキストに対するアクションのリストを返すフックを提供します。

名前	値のタイプ	任意	説明
<code>contextId</code>	<code>string</code>	いいえ	コンテキスト ID は、提供したアクションのスコープをアプリケーションの特定のエリアに限定するのに役立ちますたとえば、 <code>トポロジー</code> および <code>helm</code> などがあります。
<code>provider</code>	<code>CodeRef<ExtensionHook<Action[], any>></code>	いいえ	指定のスコープのアクションを返す React フック。 <code>contextId = resource</code> の場合には、スコープは常に Kubernetes リソースオブジェクトになります。

7.6.1.4. `console.action/resource-provider`

ResourceActionProvider は、特定のリソースモデルに対するアクションのリストを返すフックを提供します。

名前	値のタイプ	任意	説明
model	ExtensionK8sKindVersionModel	いいえ	このプロバイダーがアクションを提供するモデル。
provider	CodeRef<ExtensionHook<Action[], any>>	いいえ	指定のリソースモデルに対するアクションを返す反応フック

7.6.1.5. `console.alert-action`

このエクステンションを使用すると、特定の Prometheus アラートが `rule.name` 値に基づいてコンソールで観察された場合に、特定のアクションをトリガーできます。

名前	値のタイプ	任意	説明
alert	string	いいえ	<code>alert.rule.name</code> プロパティで定義されたアラート名
text	string	いいえ	
action	CodeRef<(alert: any) => void>	いいえ	副次的な影響を実行する関数

7.6.1.6. `console.catalog/item-filter`

このエクステンションは、特定のカタログ項目をフィルタリングできるハンドラーを追加するプラグインに使用できます。たとえばプラグインは、特定のプロバイダーからの Helm チャートをフィルタリングするフィルターを追加できます。

名前	値のタイプ	任意	説明
<code>catalogId</code>	string string[]	いいえ	このプロバイダーが提供するカタログの一意の識別子。
<code>type</code>	string	いいえ	カタログ項目タイプのタイプ ID。

名前	値のタイプ	任意	説明
filter	<code>CodeRef<(item: CatalogItem) => boolean></code>	いいえ	特定のタイプの項目をフィルタリングします。Value は、 <code>CatalogItem[]</code> を受け取り、フィルター条件に基づいてサブセットを返す関数です。

7.6.1.7. `console.catalog/item-metadata`

このエクステンションを使用すると、特定のカタログ項目に追加のメタデータを追加するプロバイダーを追加できます。

名前	値のタイプ	任意	説明
<code>catalogId</code>	<code>string string[]</code>	いいえ	このプロバイダーが提供するカタログの一意の識別子。
<code>type</code>	<code>string</code>	いいえ	カタログ項目タイプのタイプ ID。
<code>provider</code>	<code>CodeRef<ExtensionHook<CatalogItemMetadataProviderFunction, CatalogExtensionHookOptions>></code>	いいえ	特定のタイプのカタログ項目にメタデータを提供するために使用される関数を返すフック。

7.6.1.8. `console.catalog/item-provider`

このエクステンションを使用すると、プラグインはカタログ項目タイプのプロバイダーを追加できます。たとえば、Helm プラグインは、すべての Helm チャートを取得するプロバイダーを追加できます。このエクステンションを他のプラグインで使用して、特定のカタログ項目タイプをさらに追加することもできます。

名前	値のタイプ	任意	説明
<code>catalogId</code>	<code>string string[]</code>	いいえ	このプロバイダーが提供するカタログの一意の識別子。
<code>type</code>	<code>string</code>	いいえ	カタログ項目タイプのタイプ ID。

名前	値のタイプ	任意	説明
title	string	いいえ	カタログ項目プロバイダーのタイトル
provider	CodeRef<ExtensionHook<CatalogItem<any>[], CatalogExtensionHookOptions>>	いいえ	項目を取得し、これをカタログ用に正規化します。値は反応効果フックです。
priority	number	はい	このプロバイダーの優先順位。デフォルトは 0 です。優先度の高いプロバイダーは、他のプロバイダーが提供するカタログ項目を上書きする可能性があります。

7.6.1.9. `console.catalog/item-type`

このエクステンションを使用すると、プラグインはカタログ項目の新しいタイプを追加できます。たとえば Helm プラグインは、開発者カタログに追加する新しいカタログ項目タイプを HelmCharts として定義できます。

名前	値のタイプ	任意	説明
type	string	いいえ	カタログ項目をタイプ。
title	string	いいえ	カタログ項目のタイトル。
catalogDescription	string CodeRef<React.ReactNode>	はい	カタログに固有のタイプの説明。
typeDescription	string	はい	カタログ項目タイプの説明。
filters	CatalogItemAttribute[]	はい	カタログ項目に固有のカスタムフィルター。
groupings	CatalogItemAttribute[]	はい	カタログ項目に固有のカスタムグルーピング。

7.6.1.10. `console.catalog/item-type-metadata`

このエクステンションを使用すると、プラグインは任意のカタログ項目タイプのカスタムフィルターやグループ化などのメタデータを追加できます。たとえばプラグインは、チャートプロバイダーに基づきフィルタリングできる HelmCharts のカスタムフィルターをアタッチできます。

名前	値のタイプ	任意	説明
<code>type</code>	<code>string</code>	いいえ	カタログ項目をタイプ。
<code>filters</code>	<code>CatalogItemAttribute[]</code>	はい	カタログ項目に固有のカスタムフィルター。
<code>groupings</code>	<code>CatalogItemAttribute[]</code>	はい	カタログ項目に固有のカスタムグルーピング。

7.6.1.11. `console.cluster-overview/inventory-item`

新しいインベントリー項目をクラスターの概要ページに追加します。

名前	値のタイプ	任意	説明
<code>component</code>	<code>CodeRef<React.ComponentType<{}>></code>	いいえ	レンダリングされるコンポーネント。

7.6.1.12. `console.cluster-overview/multiline-utilization-item`

新しいクラスター概要のマルチライン使用状況項目を追加します。

名前	値のタイプ	任意	説明
<code>title</code>	<code>string</code>	いいえ	使用状況項目のタイトル。
<code>getUtilizationQueries</code>	<code>CodeRef<GetMultilineUtilizationQueries></code>	いいえ	Prometheus 使用状況クエリー。
<code>humanize</code>	<code>CodeRef<Humanize></code>	いいえ	Prometheus データを人間が判読できる形式に変換します。
<code>TopConsumerPopovers</code>	<code>CodeRef<React.ComponentType<TopConsumerPopoverProps>[]></code>	はい	プレーン値の代わりに Top コンシューマーポップオーバーを表示します。

7.6.1.13. `console.cluster-overview/utilization-item`

新しいクラスター概要の使用状況項目を追加します。

名前	値のタイプ	任意	説明
<code>title</code>	<code>string</code>	いいえ	使用状況項目のタイトル。
<code>getUtilizationQuery</code>	<code>CodeRef<GetQuery></code>	いいえ	Prometheus 使用状況クエリー。
<code>humanize</code>	<code>CodeRef<Humanize></code>	いいえ	Prometheus データを人間が判読できる形式に変換します。
<code>getTotalQuery</code>	<code>CodeRef<GetQuery></code>	はい	Prometheus 合計のクエリー。
<code>getRequestQuery</code>	<code>CodeRef<GetQuery></code>	はい	Prometheus 要求のクエリー。
<code>getLimitQuery</code>	<code>CodeRef<GetQuery></code>	はい	Prometheus 制限のクエリー。
<code>TopConsumerPopover</code>	<code>CodeRef<React.ComponentType<TopConsumerPopoverProps>></code>	はい	プレーン値の代わりに Top コンシューマー ポップオーバーを表示します。

7.6.1.14. `console.context-provider`

新しい React コンテキストプロバイダーを Web コンソールのアプリケーションルートに追加します。

名前	値のタイプ	任意	説明
<code>provider</code>	<code>CodeRef<Provider<T>></code>	いいえ	Context プロバイダーコンポーネント。
<code>useValueHook</code>	<code>CodeRef<() => T></code>	いいえ	コンテキスト値のフック。

7.6.1.15. `console.create-project-modal`

このエクステンションを使用すると、標準の `create` プロジェクトモーダルの代わりにレンダリングされるコンポーネントを渡すことができます。

名前	値のタイプ	任意	説明

名前	値のタイプ	任意	説明
component	CodeRef<ModalComponent<CreateProjectModalProps>>	いいえ	create プロジェクトモーダルの代わりにレンダリングするコンポーネント。

7.6.1.16. `console.dashboards/card`

新しいダッシュボードカードを追加します。

名前	値のタイプ	任意	説明
tab	string	いいえ	カードを追加するダッシュボードタブの ID。
position	'LEFT' 'RIGHT' 'MAIN'	いいえ	ダッシュボードのカードのグリッド位置。
component	CodeRef<React.ComponentType<{}>>	いいえ	ダッシュボードカードのコンポーネント。
span	OverviewCardSpan	はい	列内のカードの垂直スパン。小さな画面では無視され、デフォルトは 12 です。

7.6.1.17. `console.dashboards/custom/overview/detail/item`

Overview ダッシュボードの Details カードに項目を追加します。

名前	値のタイプ	任意	説明
title	string	いいえ	Details カードのタイトル
component	CodeRef<React.ComponentType<{}>>	いいえ	OverviewDetailItem コンポーネントによってレンダリングされる値
valueClassName	string	はい	className の値
isLoading	CodeRef<() => boolean>	はい	コンポーネントのロード中の状態を返す関数
error	CodeRef<() => string>	はい	コンポーネントごとに表示するエラーを返す関数

名前	値のタイプ	任意	説明
----	-------	----	----

7.6.1.18. `console.dashboards/overview/activity/resource`

Kubernetes リソースの監視に基づいてアクティビティをトリガーしている Overview ダッシュボードの Activity カードにアクティビティを追加します。

名前	値のタイプ	任意	説明
<code>k8sResource</code>	<code>CodeRef<FirehoseResource & { isList: true; }></code>	いいえ	置き換える使用状況項目。
<code>component</code>	<code>CodeRef<React.ComponentType<K8sActivityProps<T>>></code>	いいえ	アクションコンポーネント。
<code>isActivity</code>	<code>CodeRef<(resource: T) => boolean></code>	はい	指定のリソースがアクションを表すかどうかを判断する関数。定義されていない場合は、すべてのリソースがアクティビティを表します。
<code>getTimestamp</code>	<code>CodeRef<(resource: T) => Date></code>	はい	指定のアクションのタイムスタンプで、順序付けに使用されます。

7.6.1.19. `console.dashboards/overview/health/operator`

ステータスのソースが Kubernetes REST API である Overview ダッシュボードのステータスカードに health サブシステムを追加します。

名前	値のタイプ	任意	説明
<code>title</code>	<code>string</code>	いいえ	ポップアップメニューの Operators セクションのタイトル。
<code>resources</code>	<code>CodeRef<FirehoseResource[]></code>	いいえ	フェッチされ、 <code>healthHandler</code> に渡される Kubernetes リソース。

名前	値のタイプ	任意	説明
getOperatorsWithStatuses	CodeRef<GetOperatorsWithStatuses<T>>	はい	Operator のステータスを解決します。
operatorRowLoader	CodeRef<React.ComponentType<OperatorRowProps<T>>>	はい	ポップアップ行コンポーネントのローダー。
viewAllLink	string	はい	すべてのリソースページへのリンク。指定しない場合は、resources prop から最初のリソースのリストページが使用されます。

7.6.1.20. `console.dashboards/overview/health/prometheus`

ステータスのソースが Prometheus である Overview ダッシュボードのステータスカードに health サブシステムを追加します。

名前	値のタイプ	任意	説明
title	string	いいえ	サブシステムの表示名。
queries	string[]	いいえ	Prometheus クエリー
healthHandler	CodeRef<PrometheusHealthHandler>	いいえ	サブシステムの健全性を解決します。
additionalResource	CodeRef<FirehoseResource>	はい	フェッチされ、 healthHandler に渡される追加のリソース。
popupComponent	CodeRef<React.ComponentType<PrometheusHealthPopupProps>>	はい	ポップアップメニューコンテンツのローダー。定義された場合、health 項目はリンクとして表され、指定のコンテンツを含むポップアップメニューが開きます。
popupTitle	string	はい	ポップオーバーのタイトル。

名前	値のタイプ	任意	説明
<code>disallowedControlPlaneTopology</code>	<code>string[]</code>	はい	サブシステムを非表示にする必要のあるコントロールプレーントポロジー。

7.6.1.21. `console.dashboards/overview/health/resource`

ステータスのソースが Kubernetes リソースである概要ダッシュボードのステータスカードに health サブシステムを追加します。

名前	値のタイプ	任意	説明
<code>title</code>	<code>string</code>	いいえ	サブシステムの表示名。
<code>resources</code>	<code>CodeRef<WatchK8sResources<T>></code>	いいえ	フェッチされ、 <code>healthHandler</code> に渡される Kubernetes リソース。
<code>healthHandler</code>	<code>CodeRef<ResourceHealthHandler<T>></code>	いいえ	サブシステムの健全性を解決します。
<code>popupComponent</code>	<code>CodeRef<WatchK8sResults<T>></code>	はい	ポップアップメニューコンテンツのローダー。定義された場合、 <code>health</code> 項目はリンクとして表され、指定のコンテンツを含むポップアップメニューが開きます。
<code>popupTitle</code>	<code>string</code>	はい	ポップオーバーのタイトル。

7.6.1.22. `console.dashboards/overview/health/url`

ステータスのソースが Kubernetes REST API である概要ダッシュボードのステータスカードに health サブシステムを追加します。

名前	値のタイプ	任意	説明
<code>title</code>	<code>string</code>	いいえ	サブシステムの表示名。

名前	値のタイプ	任意	説明
url	string	いいえ	データの取得元の URL。これには、ベース Kubernetes URL が接頭辞として付けられます。
healthHandler	CodeRef<URLHealthHandler<T, K8sResourceCommon K8sResourceCommon[]>>	いいえ	サブシステムの健全性を解決します。
additionalResource	CodeRef<FirehoseResource>	はい	フェッチされ、 healthHandler に渡される追加のリソース。
popupComponent	CodeRef<React.ComponentType<{ healthResult?: T; healthResultError?: any; k8sResult?: FirehoseResult<R>; }>>	はい	ポップアップコンテンツのローダー。定義された場合、health 項目は指定のコンテンツのポップアップが開くリンクとして表示されます。
popupTitle	string	はい	ポップオーバーのタイトル。

7.6.1.23. `console.dashboards/overview/inventory/item`

概要インベントリーカードにリソーススタイルを追加します。

名前	値のタイプ	任意	説明
model	CodeRef<T>	いいえ	取得する resource のモデル。モデルの label または abbr の取得に使用します。
mapper	CodeRef<StatusGroupMapper<T, R>>	はい	さまざまなステータスをグループにマッピングする関数。
additionalResources	CodeRef<WatchK8sResources<R>>	はい	フェッチされ、 mapper 関数に渡される追加のリソース。

7.6.1.24. console.dashboards/overview/inventory/item/group

インベントリーのステータスグループを追加します。

名前	値のタイプ	任意	説明
id	string	いいえ	ステータスグループの ID。
icon	CodeRef<React.ReactElement<any, string React.JSXElementConstructor<any>>>	いいえ	ステータスグループアイコンを表す React コンポーネント。

7.6.1.25. console.dashboards/overview/inventory/item/replacement

概要のインベントリーカードを置き換えます。

名前	値のタイプ	任意	説明
model	CodeRef<T>	いいえ	取得する resource のモデル。モデルの label または abbr の取得に使用します。
mapper	CodeRef<StatusGroupMapper<T, R>>	はい	さまざまなステータスをグループにマッピングする関数。
additionalResources	CodeRef<WatchK8sResources<R>>	はい	フェッチされ、 mapper 関数に渡される追加のリソース。

7.6.1.26. console.dashboards/overview/prometheus/activity/resource

Kubernetes リソースの監視に基づいてアクティビティをトリガーしている Prometheus Overview ダッシュボードの Activity カードにアクティビティを追加します。

名前	値のタイプ	任意	説明
queries	string[]	いいえ	監視するクエリー。
component	CodeRef<React.ComponentType<PrometheusActivityProps>>	いいえ	アクションコンポーネント。

名前	値のタイプ	任意	説明
isActivity	<code>CodeRef<(results: PrometheusResponse[] => boolean></code>	はい	指定のリソースがアクションを表すかどうかを判断する関数。定義されていない場合は、すべてのリソースがアクティビティを表します。

7.6.1.27. `console.dashboards/project/overview/item`

プロジェクトの概要インベントリーカードにリソーススタイルを追加します。

名前	値のタイプ	任意	説明
model	<code>CodeRef<T></code>	いいえ	取得する resource のモデル。モデルの label または abbr の取得に使用します。
mapper	<code>CodeRef<StatusGroupMapper<T, R>></code>	はい	さまざまなステータスをグループにマッピングする関数。
additionalResources	<code>CodeRef<WatchK8sResources<R>></code>	はい	フェッチされ、 mapper 関数に渡される追加のリソース。

7.6.1.28. `console.dashboards/tab`

Overview タブの後に置かれた新規ダッシュボードタブを追加します。

名前	値のタイプ	任意	説明
id	<code>string</code>	いいえ	このタブにカードを追加する場合にタブリンク href として使用される一意のタブ ID。
navSection	<code>'home' 'storage'</code>	いいえ	タブが属するナビゲーションセクション。
title	<code>string</code>	いいえ	タブのタイトル。

7.6.1.29. `console.file-upload`

このエクステンションを使用すると、特定のファイル拡張子に対するファイルドロップアクションのハンドラーを追加できます。

名前	値のタイプ	任意	説明
<code>fileExtensions</code>	<code>string[]</code>	いいえ	サポートされるファイル拡張子。
<code>handler</code>	<code>CodeRef<FileUploadHandler></code>	いいえ	ファイルドロップアクションを処理する関数。

7.6.1.30. `console.flag`

Web コンソール機能フラグを完全に制御します。

名前	値のタイプ	任意	説明
<code>handler</code>	<code>CodeRef<FeatureFlagHandler></code>	いいえ	任意の機能フラグを設定または設定解除するのに使用されます。

7.6.1.31. `console.flag/hookProvider`

フックハンドラーを使用して Web コンソール機能フラグを完全に制御します。

名前	値のタイプ	任意	説明
<code>handler</code>	<code>CodeRef<FeatureFlagHandler></code>	いいえ	任意の機能フラグを設定または設定解除するのに使用されます。

7.6.1.32. `console.flag/model`

クラスター上の **CustomResourceDefinition** (CRD) オブジェクトの存在によって駆動される、新しい Web コンソール機能フラグを追加します。

名前	値のタイプ	任意	説明
<code>flag</code>	<code>string</code>	いいえ	CRD が検出された後に設定するフラグの名前。
<code>model</code>	<code>ExtensionK8sModel</code>	いいえ	CRD を指すモデル。

7.6.1.33. `console.global-config`

このエクステンションは、クラスターの設定を管理するために使用されるリソースを識別します。Administration → Cluster Settings → Configuration ページに、リソースへのリンクが追加されます。

名前	値のタイプ	任意	説明
id	string	いいえ	クラスター設定リソースインスタンスの一意の識別子。
name	string	いいえ	クラスター設定リソースインスタンスの名前。
model	ExtensionK8sModel	いいえ	クラスター設定リソースを参照するモデル。
namespace	string	いいえ	クラスター設定リソースインスタンスのnamespace。

7.6.1.34. console.model-metadata

API 検出で取得および生成される値を上書きして、モデルの表示をカスタマイズします。

名前	値のタイプ	任意	説明
model	ExtensionK8sGroupModel	いいえ	カスタマイズするモデル。グループのみ、またはオプションのバージョンおよび種類を指定できます。
badge	ModelBadge	はい	このモデル参照をテクノロジープレビューまたは開発者レビューとみなすかどうか。
color	string	はい	このモデルに関連付ける色。
label	string	はい	ラベルをオーバーライドします。 kind を指定する必要があります。
labelPlural	string	はい	複数形のラベルをオーバーライドします。 kind を指定する必要があります。

名前	値のタイプ	任意	説明
abbr	string	はい	省略形をカスタマイズします。デフォルトは kind のすべての大文字(最大 4 文字)です。その kind を指定する必要があります。

7.6.1.35. `console.navigation/href`

このエクステンションを使用すると、UI 内の特定のリンクを指すナビゲーション項目を追加できます。

名前	値のタイプ	任意	説明
id	string	いいえ	この項目の一意の識別子。
name	string	いいえ	この項目の名前。
href	string	いいえ	リンクの href の値。
perspective	string	はい	この項目が属するパースペクティブ ID。指定されていない場合は、デフォルトのパースペクティブに提供します。
section	string	はい	この項目が属するナビゲーションセクション。指定されていない場合は、この項目を最上位のリンクとしてレンダリングします。
dataAttributes	{ [key: string]: string; }	はい	データ属性を DOM に追加します。
startsWith	string[]	はい	URL がこのパスのいずれかで始まる場合は、この項目をアクティブと識別します。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。

名前	値のタイプ	任意	説明
<code>insertAfter</code>	<code>string string[]</code>	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 <code>insertBefore</code> が優先されます。
<code>namespaced</code>	<code>boolean</code>	はい	<code>true</code> の場合、/ns/active-namespace を最後に追加します。
<code>prefixNamespaced</code>	<code>boolean</code>	はい	<code>true</code> の場合、先頭に /k8s/ns/active-namespace が追加されます。

7.6.1.36. `console.navigation/resource-cluster`

このエクステンションを使用すると、クラスターリソースの詳細ページを指すナビゲーションアイテムを追加できます。そのリソースの K8s モデルを使用して、ナビゲーション項目を定義できます。

名前	値のタイプ	任意	説明
<code>id</code>	<code>string</code>	いいえ	この項目の一意の識別子。
<code>model</code>	<code>ExtensionK8sModel</code>	いいえ	このナビゲーション項目がリンクするモデル。
<code>perspective</code>	<code>string</code>	はい	この項目が属するペースペクティブ ID。指定されていない場合は、デフォルトのペースペクティブに提供します。
<code>section</code>	<code>string</code>	はい	この項目が属するナビゲーションセクション。指定しない場合は、この項目をトップレベルのリンクとしてレンダリングします。
<code>dataAttributes</code>	<code>{ [key: string]: string; }</code>	はい	データ属性を DOM に追加します。

名前	値のタイプ	任意	説明
startsWith	string[]	はい	URL がこのパスのいずれかで始まる場合は、この項目をアクティブと識別します。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
insertAfter	string string[]	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 insertBefore が優先されます。
name	string	はい	デフォルト名をオーバーライドします。指定されていない場合、リンクの名前はモデルの複数形の値と同じになります。

7.6.1.37. `console.navigation/resource-ns`

このエクステンションを使用すると、namespaced リソースの詳細ページを指すナビゲーション項目を追加できます。そのリソースの K8s モデルを使用して、ナビゲーション項目を定義できます。

名前	値のタイプ	任意	説明
id	string	いいえ	この項目の一意の識別子。
model	ExtensionK8sModel	いいえ	このナビゲーション項目がリンクするモデル。
perspective	string	はい	この項目が属するペースペクティブ ID。指定されていない場合は、デフォルトのペースペクティブに提供します。

名前	値のタイプ	任意	説明
section	string	はい	この項目が属するナビゲーションセクション。指定しない場合は、この項目をトップレベルのリンクとしてレンダリングします。
dataAttributes	{ [key: string]: string; }	はい	データ属性を DOM に追加します。
startsWith	string[]	はい	URL がこのパスのいずれかで始まる場合は、この項目をアクティブと識別します。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
insertAfter	string string[]	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 insertBefore が優先されます。
name	string	はい	デフォルト名をオーバーライドします。指定されていない場合、リンクの名前はモデルの複数形の値と同じになります。

7.6.1.38. `console.navigation/section`

このエクステンションを使用すると、ナビゲーションタブ内の新しいナビゲーション項目セクションを定義できます。

名前	値のタイプ	任意	説明
id	string	いいえ	この項目の一意の識別子。

名前	値のタイプ	任意	説明
perspective	string	はい	この項目が属するペースペクティブ ID。指定されていない場合は、デフォルトのペースペクティブに提供します。
dataAttributes	{ [key: string]: string; }	はい	データ属性を DOM に追加します。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
insertAfter	string string[]	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 insertBefore が優先されます。
name	string	はい	このセクションの名前。指定しない場合は、セクションの上に区切り記号のみが表示されます。

7.6.1.39. `console.navigation/separator`

このエクステンションを使用すると、ナビゲーション内のナビゲーション項目間に区切り文字を追加できます。

名前	値のタイプ	任意	説明
id	string	いいえ	この項目の一意の識別子。
perspective	string	はい	この項目が属するペースペクティブ ID。指定されていない場合は、デフォルトのペースペクティブに提供します。

名前	値のタイプ	任意	説明
section	string	はい	この項目が属するナビゲーションセクション。指定されていない場合は、この項目を最上位のリンクとしてレンダリングします。
dataAttributes	{ [key: string]: string; }	はい	データ属性を DOM に追加します。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
insertAfter	string string[]	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 insertBefore が優先されます。

7.6.1.40. `console.page/resource/details`

名前	値のタイプ	任意	説明
model	ExtensionK8sGroup KindModel	いいえ	このリソースページがリンクするモデル。
component	CodeRef<React.ComponentType<{ match: match<{}>; namespace: string; model: ExtensionK8sModel; }>>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。

7.6.1.41. `console.page/resource/list`

Console ルーターに新しいリソースリストのページを追加します。

名前	値のタイプ	任意	説明
model	ExtensionK8sGroupKindModel	いいえ	このリソースページがリンクするモデル。
component	CodeRef<React.ComponentType<{ match: match<{}>; namespace: string; model: ExtensionK8sModel; }>>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。

7.6.1.42. `console.page/route`

Web コンソールルーターに新しいページを追加します。 [React Router](#) を参照してください。

名前	値のタイプ	任意	説明
component	CodeRef<React.ComponentType<RouteComponentProps<{}, StaticContext, any>>>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。
path	string string[]	いいえ	path-to-regexp@^1.7.0 が理解する有効な URL パスまたはパスの配列。
perspective	string	はい	このページが属するペースペクティブ。 指定されていない場合は、すべてのペースペクティブに提供します。
exact	boolean	はい	true の場合、パスが location.pathname と完全に一致する場合にのみマッチします。

7.6.1.43. `console.page/route/standalone`

一般的なページレイアウトの外部でレンダリングされる新しいスタンドアロンページを Web コンソールルーターに追加します。 [React Router](#) を参照してください。

名前	値のタイプ	任意	説明
<code>component</code>	<code>CodeRef<React.ComponentType<RouteComponentProps<{}, StaticContext, any>>></code>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。
<code>path</code>	<code>string string[]</code>	いいえ	<code>path-to-regexp@^1.7.0</code> が理解する有効な URL パスまたはパスの配列。
<code>exact</code>	<code>boolean</code>	はい	<code>true</code> の場合、パスが <code>location.pathname</code> と完全に一致する場合にのみマッチします。

7.6.1.44. `console.perspective`

このエクステンションを使用すると、コンソールに新しいパースペクティブを追加してナビゲーションメニューをカスタマイズできます。

名前	値のタイプ	任意	説明
<code>id</code>	<code>string</code>	いいえ	パースペクティブの識別子。
<code>name</code>	<code>string</code>	いいえ	パースペクティブの表示名。
<code>icon</code>	<code>CodeRef<LazyComponent></code>	いいえ	パースペクティブの表示アイコン。
<code>landingPageURL</code>	<code>CodeRef<(flags: { [key: string]: boolean; }, isFirstVisit: boolean) => string></code>	いいえ	パースペクティブのランディングページの URL を取得する関数。
<code>importRedirectURL</code>	<code>CodeRef<(namespace: string) => string></code>	いいえ	インポートフローのリダイレクト URL を取得する関数。
<code>default</code>	<code>boolean</code>	はい	パースペクティブがデフォルトであるかどうか。デフォルトは1つのみです。

名前	値のタイプ	任意	説明
defaultPins	ExtensionK8sModel[]	はい	ナビゲーション上のデフォルトの固定されたリソース
usePerspectiveDetection	CodeRef<() ⇒ [boolean, boolean]>	はい	デフォルトのペースペクティブを検出するフック

7.6.1.45. `console.project-overview/inventory-item`

新しいインベントリー項目を プロジェクトの概要 ページに追加します。

名前	値のタイプ	任意	説明
component	CodeRef<React.ComponentType<{ projectName: string; }>>	いいえ	レンダリングされるコンポーネント。

7.6.1.46. `console.project-overview/utilization-item`

新しいプロジェクト概要の使用状況項目を追加します。

名前	値のタイプ	任意	説明
title	string	いいえ	使用状況項目のタイトル。
getUtilizationQuery	CodeRef<GetProjectQuery>	いいえ	Prometheus 使用状況クエリー。
humanize	CodeRef<Humanize>	いいえ	Prometheus データを人間が判読できる形式に変換します。
getTotalQuery	CodeRef<GetProjectQuery>	はい	Prometheus 合計のクエリー。
getRequestQuery	CodeRef<GetProjectQuery>	はい	Prometheus 要求のクエリー。
getLimitQuery	CodeRef<GetProjectQuery>	はい	Prometheus 制限のクエリー。

名前	値のタイプ	任意	説明
TopConsumerPopover	CodeRef<React.ComponentType<TopConsumerPopoverProps>>	はい	プレーン値の代わりに最上位のコンシューマー ポップオーバーを表示します。

7.6.1.47. console.pvc/alert

このエクステンションを使用すると、PVC 詳細ページにカスタムアラートを追加できます。

名前	値のタイプ	任意	説明
alert	CodeRef<React.ComponentType<{ pvc: K8sResourceCommon; }>>	いいえ	アラートコンポーネント。

7.6.1.48. console.pvc/create-prop

このエクステンションを使用すると、PVC リストページで PVC リソースを作成する際に使用される追加のプロパティを指定できます。

名前	値のタイプ	任意	説明
label	string	いいえ	prop アクション作成のラベル。
path	string	いいえ	prop アクション作成のパス。

7.6.1.49. console.pvc/delete

このエクステンションを使用すると、PVC リソースの削除をフッキングできます。追加情報とカスタム PVC 削除ロジックを含むアラートを追加できます。

名前	値のタイプ	任意	説明
predicate	CodeRef<(pvc: K8sResourceCommon) => boolean>	いいえ	エクステンションを使用するかどうかを示す述語。

名前	値のタイプ	任意	説明
onPVCKill	CodeRef<(pvc: K8sResourceCommon) => Promise<void>>	いいえ	PVC 削除操作の方法。
alert	CodeRef<React.ComponentType<{ pvc: K8sResourceCommon; }>>	いいえ	追加情報を表示するアラートコンポーネント。

7.6.1.50. console.pvc/status

名前	値のタイプ	任意	説明
priority	number	いいえ	status コンポーネントの優先度。値が大きいほど優先度が高くなります。
status	CodeRef<React.ComponentType<{ pvc: K8sResourceCommon; }>>	いいえ	status コンポーネント。
predicate	CodeRef<(pvc: K8sResourceCommon) => boolean>	いいえ	ステータスコンポーネントをレンダリングするかどうかを示す述語。

7.6.1.51. console.redux-reducer

plugins.<scope> サブ状態で動作する Console Redux ストアに新しい reducer を追加します。

名前	値のタイプ	任意	説明
scope	string	いいえ	Redux 状態オブジェクト内の reducer が管理するサブ状態を表すキー。
reducer	CodeRef<Reducer<any, AnyAction>>	いいえ	reducer が管理するサブ状態で動作する reducer 関数

7.6.1.52. console.resource/create

このエクステンションを使用すると、プラグインは、ユーザーが新しいリソースインスタンスを作成しようとしたときにレンダリングされる特定のリソースのカスタムコンポーネント(つまりウィザードやフォーム)を追加できます。

名前	値のタイプ	任意	説明
model	ExtensionK8sModel	いいえ	この create resource ページがレンダリングされるモデル。
component	CodeRef<React.ComponentType<CreateResourceComponentProps>>	いいえ	モデルがマッチする場合にレンダリングされるコンポーネント

7.6.1.53. `console.resource/details-item`

詳細ページのデフォルトのリソース概要に、新しい詳細項目を追加します。

名前	値のタイプ	任意	説明
model	ExtensionK8sModel	いいえ	対象リソースの API グループ、バージョン、カインド。
id	string	いいえ	一意の ID
column	DetailsItemColumn	いいえ	項目を、詳細ページのリソース概要の '左列' と '右列' のどちらに表示するかを指定します。デフォルト: 'right'
title	string	いいえ	詳細項目のタイトル。
path	string	はい	詳細項目の値として使用されるリソースプロパティへの完全修飾パス(オプション)。primitive type の値以外は直接レンダリングできません。他のデータ型を処理するには、コンポーネントプロパティーを使用します。

名前	値のタイプ	任意	説明
component	CodeRef<React.ComponentType<DetailsItem>> ComponentProps<K8sResourceCommon, any>>>	はい	詳細項目の値をレンダリングする React コンポーネント (オプション)。
sortWeight	number	はい	同じ列内の他の詳細項目すべてに対する相対的な並べ替えの重み (オプション)。任意の有効な JavaScriptNumber で表されます。各列の項目は、低いものから高いものへと個別に並べ替えられます。並べ替えの重みがない項目は、並べ替えの重みがある項目の後に表示されます。

7.6.1.54. `console.storage-class/provisioner`

ストレージクラスの作成時に、新しいストレージクラスプロビジョナーをオプションとして追加します。

名前	値のタイプ	任意	説明
CSI	ProvisionerDetails	はい	Container Storage Interface プロビジョナータイプ
OTHERS	ProvisionerDetails	はい	Other プロビジョナータイプ

7.6.1.55. `console.storage-provider`

このエクステンションを使用すると、ストレージおよびプロバイダー固有のコンポーネントをアタッチする際に、新しいストレージプロバイダーを追加できます。

名前	値のタイプ	任意	説明
name	string	いいえ	プロバイダーの表示名。

名前	値のタイプ	任意	説明
コンポーネント	<code>CodeRef<React.ComponentType<Partial<RouteComponentProps<{}, StaticContext, any>>></code>	いいえ	レンダリングするプロバイダー固有のコンポーネント。

7.6.1.56. `console.tab`

水平ナビゲーションに、`contextId` に一致するタブを追加します。

名前	値のタイプ	任意	説明
<code>contextId</code>	<code>string</code>	いいえ	タブが挿入される水平ナビゲーションに割り当てられるコンテキスト ID。使用できる値: dev-console-observe
<code>name</code>	<code>string</code>	いいえ	タブの表示ラベル
<code>href</code>	<code>string</code>	いいえ	既存の URL に追加される <code>href</code>
<code>component</code>	<code>CodeRef<React.ComponentType<PageComponentProps<K8sResourceCommon>>></code>	いいえ	タブコンテンツのコンポーネント。

7.6.1.57. `console.tab/horizontalNav`

このエクステンションを使用すると、リソースの詳細ページにタブを追加できます。

名前	値のタイプ	任意	説明
<code>model</code>	<code>ExtensionK8sKindVersionModel</code>	いいえ	このプロバイダーがタブを表示するモデル。
<code>page</code>	<code>{ name: string; href: string; }</code>	いいえ	水平タブに表示されるページ。名前としてタブ名およびタブの <code>href</code> を取ります。

名前	値のタイプ	任意	説明
component	<code>CodeRef<React.ComponentType<PageComponentProps<K8sResourceCommon>>></code>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。

7.6.1.58. `console.telemetry/listener`

このコンポーネントは、テレメトリーイベントを受信するリスナー関数を登録するために使用できます。これらのイベントには、ユーザー識別、ページナビゲーション、その他のアプリケーション固有のイベントが含まれます。リスナーは、このデータをレポートと分析のために使用できます。

名前	値のタイプ	任意	説明
listener	<code>CodeRef<TelemetryEventListener></code>	いいえ	テレメトリーイベントをリッスンします

7.6.1.59. `console.topology/adapter/build`

`BuildAdapter` は、Build コンポーネントで使用できるデータに要素を適応させるアダプターを追加します。

名前	値のタイプ	任意	説明
adapt	<code>CodeRef<(element: GraphElement) => AdapterDataType<BuildConfigData> undefined></code>	いいえ	Build コンポーネントで使用できるデータに要素を適応させるアダプター。

7.6.1.60. `console.topology/adapter/network`

`NetworkAdapter` は、Networking コンポーネントで使用できるデータに要素を適応させるアダプターを提供します。

名前	値のタイプ	任意	説明
adapt	<code>CodeRef<(element: GraphElement) => NetworkAdapterType undefined></code>	いいえ	Networking コンポーネントで使用できるデータに要素を適応させるアダプター。

7.6.1.61. console.topology/adapter/pod

PodAdapter はアダプターを提供し、Pod コンポーネントで使用できるデータに要素を適合させます。

名前	値のタイプ	任意	説明
adapt	<code>CodeRef<element: GraphElement) => AdapterDataType<PodsAdapterDataType> undefined></code>	いいえ	Pod コンポーネントで使用できるデータに要素を適応させるアダプター。

7.6.1.62. console.topology/component/factory

ViewComponentFactory の Getter。

名前	値のタイプ	任意	説明
getFactory	<code>CodeRef<ViewComponentFactory></code>	いいえ	ViewComponentFactory の Getter。

7.6.1.63. console.topology/create/connector

コネクター作成関数の getter。

名前	値のタイプ	任意	説明
getCreateConnector	<code>CodeRef<CreateConnectionGetter></code>	いいえ	コネクター作成関数の getter。

7.6.1.64. console.topology/data/factory

トポロジーデータモデルファクトリーエクステンション

名前	値のタイプ	任意	説明
id	<code>string</code>	いいえ	ファクトリーの一意の ID。
priority	<code>number</code>	いいえ	ファクトリーの優先度
resources	<code>WatchK8sResourcesGeneric</code>	はい	<code>useK8sWatchResources</code> フックから取得されるリソース。

名前	値のタイプ	任意	説明
<code>workloadKeys</code>	<code>string[]</code>	はい	ワークロードが含まれるリソースのキー。
<code>getDataModel</code>	<code>CodeRef<TopologyDataModelGetter></code>	はい	データモデルファクトリーの Getter。
<code>isResourceDepicted</code>	<code>CodeRef<TopologyDataModelDepicted></code>	はい	リソースがこのモデルファクトリーによって記述されているかどうかを判断する関数の Getter。
<code>getDataModelReconciler</code>	<code>CodeRef<TopologyDataModelReconciler></code>	はい	すべてのエクステンションのモデルがロードされた後にデータモデルを調整する関数の Getter。

7.6.1.65. `console.topology/decorator/provider`

トポジーデコレータープロバイダーエクステンション

名前	値のタイプ	任意	説明
<code>id</code>	<code>string</code>	いいえ	エクステンション固有のトポジーデコレーターの ID
<code>priority</code>	<code>number</code>	いいえ	エクステンション固有のトポジーデコレーターの優先順位
<code>quadrant</code>	<code>TopologyQuadrant</code>	いいえ	エクステンション固有のトポジーデコレーターのクアドラント
<code>decorator</code>	<code>CodeRef<TopologyDecoratorGetter></code>	いいえ	エクステンション固有のデコレーター

7.6.1.66. `console.topology/details/resource-alert`

`DetailsResourceAlert` は、特定のトポジーコンテキストまたはグラフ要素のアラートを提供します。

名前	値のタイプ	任意	説明

名前	値のタイプ	任意	説明
id	string	いいえ	このアラートの ID。アラートの破棄後に表示しない場合に状態を保存するため使用されます。
contentProvider	CodeRef<(element: GraphElement) => DetailsResourceAlertContent null>	いいえ	アラートの内容を返すフック。

7.6.1.67. `console.topology/details/resource-link`

DetailsResourceLink は、特定のトポロジーコンテキストまたはグラフ要素のリンクを提供します。

名前	値のタイプ	任意	説明
link	CodeRef<(element: GraphElement) => React.Component undefined>	いいえ	指定された場合はリソースリンクを返し、指定されない場合は未定義を返します。スタイルには ResourceIcon および ResourceLink プロパティーを使用します。
priority	number	はい	優先度の高いファクトリーからリンクを作成します。

7.6.1.68. `console.topology/details/tab`

DetailsTab は、トポロジーの詳細パネルのタブを提供します。

名前	値のタイプ	任意	説明
id	string	いいえ	この詳細タブの一意の識別子。
label	string	いいえ	UI に表示されるタブのラベル。

名前	値のタイプ	任意	説明
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。
insertAfter	string string[]	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 insertBefore 値が優先されます。

7.6.1.69. `console.topology/details/tab-section`

DetailsTabSection は、トポロジーの詳細パネルの特定タブのセクションを提供します。

名前	値のタイプ	任意	説明
id	string	いいえ	この詳細タブセクションの一意の識別子。
tab	string	いいえ	このセクションが提供する必要のある親タブID。
provider	CodeRef<DetailsTabSectionExtensionHook>	いいえ	コンポーネントを返すフック、または null が未定義の場合、トポロジーサイドバーにレンダリングされます。SDK コンポーネント: <Section title=>...> パディング領域
section	CodeRef<(element: GraphElement, renderNull?: () => null) => React.Component undefined>	いいえ	非推奨: プロバイダーが定義されていない場合はフォールバックします。renderNull はすでに no-op です。
insertBefore	string string[]	はい	ここで参照される項目の前に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。

名前	値のタイプ	任意	説明
<code>insertAfter</code>	<code>string string[]</code>	はい	ここで参照される項目の後に、この項目を挿入します。配列の場合は、最初に見つかったものが順番に使用されます。 <code>insertBefore</code> 値が優先されます。

7.6.1.70. `console.topology/display/filters`

トポロジー表示フィルターエクステンション

名前	値のタイプ	任意	説明
<code>getTopologyFilters</code>	<code>CodeRef<() => TopologyDisplayOption[]></code>	いいえ	エクステンション固有のトポロジーフィルターのゲッター
<code>applyDisplayOptions</code>	<code>CodeRef<TopologyApplyDisplayOptions></code>	いいえ	モデルにフィルターを適用する関数

7.6.1.71. `console.topology/relationship/provider`

トポロジー関係プロバイダーコネクターエクステンション

名前	値のタイプ	任意	説明
<code>provides</code>	<code>CodeRef<RelationshipProviderProvides></code>	いいえ	ソースノードとターゲットノード間に接続を作成できるか判断するために使用
ヒント	<code>string</code>	いいえ	コネクター操作がドロップターゲット上に移動したときに表示されるツールヒント(例: "Create a Visual Connector")
<code>create</code>	<code>CodeRef<RelationshipProviderCreate></code>	いいえ	接続を作成するためにコネクターがターゲットノード上にドロップされると実行されるコールバック

名前	値のタイプ	任意	説明
priority	number	いいえ	関係の優先順位。複数の場合は高い方が優先されます

7.6.1.72. `console.user-preference/group`

このエクステンションを使用して、`console user-preferences` ページにグループを追加できます。`console user-preferences` ページの垂直タブのオプションとして表示されます。

名前	値のタイプ	任意	説明
id	string	いいえ	ユーザー設定グループを識別するのに使用される ID。
label	string	いいえ	ユーザー設定グループのラベル
insertBefore	string	はい	このユーザー設定グループの後に配置しなければならないグループの ID
insertAfter	string	はい	このユーザー設定グループの前に配置しなければならないグループの ID

7.6.1.73. `console.user-preference/item`

このエクステンションを使用して、`console user-preferences` ページのユーザー設定グループに項目を追加できます。

名前	値のタイプ	任意	説明
id	string	いいえ	ユーザー設定項目を特定するために使用され、項目の順序を定義するために <code>insertAfter</code> および <code>insertBefore</code> で参照される ID
label	string	いいえ	ユーザー設定のラベル
description	string	いいえ	ユーザー設定の説明

名前	値のタイプ	任意	説明
field	UserPreferenceField	いいえ	ユーザー設定を定義するために値をレンダリングするために使用される入力フィールドのオプション
groupId	string	はい	項目が属するユーザー優先グループを識別するために使用される ID
insertBefore	string	はい	このユーザー設定項目の後に配置しなければならない項目の ID
insertAfter	string	はい	このユーザー設定項目の前に配置しなければならない項目の ID

7.6.1.74. console.yaml-template

yaml エディターを使用してリソースを編集するための YAML テンプレート。

名前	値のタイプ	任意	説明
model	ExtensionK8sModel	いいえ	テンプレートに関連付けられたモデル。
template	CodeRef<string>	いいえ	YAML テンプレート。
name	string	いいえ	テンプレートの名前。名前 default を使用して、これをデフォルトテンプレートと識別します。

7.6.1.75. dev-console.add/action

このエクステンションを使用すると、プラグインは開発者パースペクティブの add ページに追加アクション項目を追加できます。たとえば、Serverless プラグインは、開発者コンソールの add ページにサーバーレス関数の新しい追加項目を追加できます。

名前	値のタイプ	任意	説明
id	string	いいえ	アクションを識別するための ID。
label	string	いいえ	アクションのラベル。

名前	値のタイプ	任意	説明
description	string	いいえ	アクションの説明。
href	string	いいえ	移動先の href 。
groupId	string	はい	アクションが属するアクショングループを識別するのに使用される ID。
icon	CodeRef<React.ReactNode>	はい	パースペクティブの表示アイコン。
accessReview	AccessReviewResourceAttributes[]	はい	アクションの可視性または有効化を制御するオプションのアクセスレビュー。

7.6.1.76. dev-console.add/action-group

この拡張機能を使用すると、プラグインは開発者コンソールの add ページにグループを追加できます。グループはアクションが参照でき、アクションはエクステンションの定義に基づき add action ページでグループ化されます。たとえば、Serverless プラグインは、Serverless グループと複数の追加アクションを追加できます。

名前	値のタイプ	任意	説明
id	string	いいえ	アクショングループを識別するために使用される ID
name	string	いいえ	アクショングループのタイトル
insertBefore	string	はい	このアクショングループの後に配置しなければならないグループの ID
insertAfter	string	はい	このアクショングループの前に配置しなければならないグループの ID

7.6.1.77. dev-console.import/environment

このエクステンションを使用すると、開発者コンソール git インポートフォームのビルダーイメージセレクターで追加のビルト環境変数フィールドを指定できます。これを設定すると、フィールドはビルトセクション内の同じ名前の環境変数をオーバーライドします。

名前	値のタイプ	任意	説明
<code>imageStreamName</code>	<code>string</code>	いいえ	カスタム環境変数を指定するイメージストリームの名前
<code>imageStreamTags</code>	<code>string[]</code>	いいえ	サポートされるイメージストリームタグのリスト
<code>environments</code>	<code>ImageEnvironment[]</code>	いいえ	環境変数のリスト

7.6.1.78. `console.dashboards/overview/detail/item`

非推奨: 代わりに `CustomOverviewDetailItem` タイプを使用してください。

名前	値のタイプ	任意	説明
<code>component</code>	<code>CodeRef<React.ComponentType<{}>>></code>	いいえ	<code>DetailItem</code> コンポーネントに基づく値

7.6.1.79. `console.page/resource/tab`

非推奨: 代わりに `console.tab/horizontalNav` を使用してください。Console ルーターに新しいリソースタブページを追加します。

名前	値のタイプ	任意	説明
<code>model</code>	<code>ExtensionK8sGroupKindModel</code>	いいえ	このリソースページがリンクするモデル。
<code>component</code>	<code>CodeRef<React.ComponentType<RouteComponentProps<{}, StaticContext, any>>></code>	いいえ	ルートがマッチしたときにレンダリングされるコンポーネント。
<code>name</code>	<code>string</code>	いいえ	タブの名前。
<code>href</code>	<code>string</code>	はい	タブリンクのオプション <code>href</code> 。指定しない場合は、最初の <code>path</code> が使用されます。
<code>exact</code>	<code>boolean</code>	はい	<code>true</code> の場合、パスが <code>location.pathname</code> と完全に一致する場合にのみマッチします。

7.6.2. 動的プラグイン API

7.6.2.1. useActivePerspective

現在アクティブなパースペクティブとアクティブなパースペクティブを設定するためのコールバックを提供するフック。現在アクティブなパースペクティブとセッターコールバックを含むタプルを返します。

例

```
const Component: React.FC = (props) => {
  const [activePerspective, setActivePerspective] = useActivePerspective();
  return <select
    value={activePerspective}
    onChange={(e) => setActivePerspective(e.target.value)}
  >
    {
      // ...perspective options
    }
  </select>
}
```

7.6.2.2. GreenCheckCircleIcon

緑色のチェックマークの円形アイコンを表示するためのコンポーネント。

例

```
<GreenCheckCircleIcon title="Healthy" />
```

パラメータ名	説明
className	(オプション) コンポーネントの追加クラス名
title	(オプション) アイコンのタイトル
size	(オプション) アイコンのサイズ: (sm 、 md 、 lg 、 xl)

7.6.2.3. RedExclamationCircleIcon

赤い感嘆符の円形アイコンを表示するためのコンポーネント。

例

```
<RedExclamationCircleIcon title="Failed" />
```

パラメータ名	説明

パラメーター名	説明
className	(オプション) コンポーネントの追加クラス名
title	(オプション) アイコンのタイトル
size	(オプション) アイコンのサイズ: (sm、 md、 lg、 xl)

7.6.2.4. YellowExclamationTriangleIcon

黄色の三角形の感嘆符アイコンを表示するためのコンポーネント。

例

```
<YellowExclamationTriangleIcon title="Warning" />
```

パラメーター名	説明
className	(オプション) コンポーネントの追加クラス名
title	(オプション) アイコンのタイトル
size	(オプション) アイコンのサイズ: (sm、 md、 lg、 xl)

7.6.2.5. BlueInfoCircleIcon

青い情報円形アイコンを表示するためのコンポーネント。

例

```
<BlueInfoCircleIcon title="Info" />
```

パラメーター名	説明
className	(オプション) コンポーネントの追加クラス名
title	(オプション) アイコンのタイトル
size	(オプション) アイコンのサイズ: ('sm'、 'md'、 'lg'、 'xl')

7.6.2.6. ErrorStatus

エラーステータスのポップオーバーを表示するためのコンポーネント。

例

```
<ErrorStatus title={errorMsg} />
```

パラメータ名	説明
title	(オプション) ステータステキスト
iconOnly	(オプション) true の場合、アイコンのみを表示します
noTooltip	(オプション) true の場合、ツールチップは表示されません
className	(オプション) コンポーネントの追加クラス名
popoverTitle	(オプション) ポップオーバーのタイトル

7.6.2.7. InfoStatus

情報ステータスのポップオーバーを表示するためのコンポーネント。

例

```
<InfoStatus title={infoMsg} />
```

パラメータ名	説明
title	(オプション) ステータステキスト
iconOnly	(オプション) true の場合、アイコンのみを表示します
noTooltip	(オプション) true の場合、ツールチップは表示されません
className	(オプション) コンポーネントの追加クラス名
popoverTitle	(オプション) ポップオーバーのタイトル

7.6.2.8. ProgressStatus

進行状況のポップオーバーを表示するためのコンポーネント。

例

```
<ProgressStatus title={progressMsg} />
```

パラメーターネーム	説明
title	(オプション) ステータステキスト
iconOnly	(オプション) true の場合、アイコンのみを表示します
noTooltip	(オプション) true の場合、ツールチップは表示されません
className	(オプション) コンポーネントの追加クラス名
popoverTitle	(オプション) ポップオーバーのタイトル

7.6.2.9. SuccessStatus

成功ステータスのポップオーバーを表示するためのコンポーネント。

例

```
<SuccessStatus title={successMsg} />
```

パラメーターネーム	説明
title	(オプション) ステータステキスト
iconOnly	(オプション) true の場合、アイコンのみを表示します
noTooltip	(オプション) true の場合、ツールチップは表示されません
className	(オプション) コンポーネントの追加クラス名
popoverTitle	(オプション) ポップオーバーのタイトル

7.6.2.10. checkAccess

特定のリソースへのユーザーアクセスに関する情報を提供します。リソースアクセス情報を含むオブジェクトを返します。

パラメーターネーム	説明
resourceAttributes	アクセスレビューのリソース属性
impersonate	権限借用の詳細

7.6.2.11. useAccessReview

特定のリソースへのユーザーアクセスに関する情報を提供するフック。**isAllowed** と **loading** 値を含む配列を返します。

パラメーター名	説明
resourceAttributes	アクセスレビューのリソース属性
impersonate	権限借用の詳細

7.6.2.12. useResolvedExtensions

解決された **CodeRef** プロパティーで Console 拡張機能を使用するための React フック。このフックは、**useExtensions** フックと同じ引数を受け入れ、拡張インスタンスの適合したリストを返し、各拡張のプロパティー内のすべてのコード参照を解決します。

最初に、フックは空の配列を返します。解決が完了すると、React コンポーネントが再レンダリングされ、適合した拡張機能のリストが返されます。一致する拡張子のリストが変更されると、解決が再開されます。フックは解決が完了するまで前の結果を返し続けます。

フックの結果要素は、再レンダリング全体で参照的に安定していることが保証されています。解決されたコード参照、解決が完了したかどうかを示す布尔値フラグ、および解決中に検出されたエラーのリストを含む適応拡張インスタンスのリストを含むタプルを返します。

例

```
const [navItemExtensions, navItemsResolved] = useResolvedExtensions<NavItem>(isNavItem);
// process adapted extensions and render your component
```

パラメーター名	説明
typeGuards	それぞれが動的プラグイン拡張機能を引数として受け入れ、拡張機能が目的の型制約を満たしているかどうかを示す布尔値フラグを返すコールバックのリスト

7.6.2.13. HorizontalNav

ページのナビゲーションバーを作成するコンポーネント。ルーティングはコンポーネントの一部として処理されます。**console.tab/HorizontalNav** を使用すると、水平ナビゲーションにコンテンツを追加できます。

例

```
const HomePage: React.FC = (props) => {
  const page = {
    href: 'home',
    name: 'Home',
    component: () => <>Home</>
```

```

    }
    return <HorizontalNav match={props.match} pages={[page]} />
  }
}

```

パラメータ名	説明
resource	K8sResourceCommon タイプのオブジェクトである、このナビゲーションに関連付けられたリソース
pages	ページオブジェクトの配列
match	React Router が提供する match オブジェクト

7.6.2.14. TableData

テーブル行内にテーブルデータを表示するためのコンポーネント。

例

```

const PodRow: React.FC<RowProps<K8sResourceCommon>> = ({ obj, activeColumnIDs }) => {
  return (
    <>
      <TableData id={columns[0].id} activeColumnIDs={activeColumnIDs}>
        <ResourceLink kind="Pod" name={obj.metadata.name} namespace={obj.metadata.namespace}>
      />
      </TableData>
      <TableData id={columns[1].id} activeColumnIDs={activeColumnIDs}>
        <ResourceLink kind="Namespace" name={obj.metadata.namespace}>
      </TableData>
    </>
  );
};

```

パラメータ名	説明
id	テーブルの一意の ID
activeColumnIDs	アクティブな列
className	(オプション) スタイリングのオプションクラス名

7.6.2.15. useActiveColumns

ユーザーが選択したアクティブな TableColumns のリストを提供するフック。

例

```

// See implementation for more details on TableColumn type
const [activeColumns, userSettingsLoaded] = useActiveColumns({

```

```

    columns,
    showNamespaceOverride: false,
    columnManagementID,
  });
  return userSettingsAreLoaded ? <VirtualizedTable columns={activeColumns} {...otherProps} /> : null

```

パラメーターネーム	説明
options	キーと値のマップとして渡されるもの。
\{ TableColumn[] \} options.columns	使用可能なすべての TableColumn の配列
{boolean} [options.showNamespaceOverride]	(オプション) true の場合、列管理の選択に関係なく、namespace 列が含まれます。
{string} [options.columnManagementID]	(オプション) ユーザー設定との間で列管理の選択を保持および取得するために使用される一意の ID。通常は、リソースのグループ/バージョン/種類 (GVK) の文字列です。

現在のユーザーが選択したアクティブな列 (options.columns のサブセット) と、ユーザー設定がロードされたかどうかを示すブール値フラグを含むタプル。

7.6.2.16. ListPageHeader

ページヘッダーを生成するためのコンポーネント。

例

```

const exampleList: React.FC = () => {
  return (
    <>
      <ListPageHeader title="Example List Page"/>
    </>
  );
};

```

パラメーターネーム	説明
title	見出しタイトル
helpText	(オプション) 反応ノードとしてのヘルプセクション
badge	(オプション) 反応ノードとしてのバッジアイコン

7.6.2.17. ListPageCreate

特定のリソースの種類に対して、そのリソースの作成用 YAML へのリンクを自動的に生成する作成ボタンを追加するためのコンポーネント。

例

```
const exampleList: React.FC<MyProps> = () => {
  return (
    <>
      <ListPageHeader title="Example Pod List Page"/>
      <ListPageCreate groupVersionKind="Pod">Create Pod</ListPageCreate>
    </ListPageHeader>
    </>
  );
};
```

パラメータ名	説明
groupVersionKind	表すためのリソースグループ/バージョン/種類

7.6.2.18. ListPageCreateLink

定型化されたリンクを作成するためのコンポーネント。

例

```
const exampleList: React.FC<MyProps> = () => {
  return (
    <>
      <ListPageHeader title="Example Pod List Page"/>
      <ListPageCreateLink to={'/link/to/my/page'}>Create Item</ListPageCreateLink>
    </ListPageHeader>
    </>
  );
};
```

パラメータ名	説明
to	リンク先の文字列の場所
createAccessReview	(オプション) アクセスを決定するために使用される namespace と種類を持つオブジェクト
children	(オプション) コンポーネントの子

7.6.2.19. ListPageCreateButton

ボタンを作成するためのコンポーネント。

例

```
const exampleList: React.FC<MyProps> = () => {
  return (
    <>
      <ListPageHeader title="Example Pod List Page"/>
      <ListPageCreateButton createAccessReview={access}>Create Pod</ListPageCreateButton>
    </ListPageHeader>
    </>
  );
};
```

パラメータ名	説明
createAccessReview	(オプション) アクセスを決定するために使用される namespace と種類を持つオブジェクト
pfButtonProps	(オプション) Patternfly Button のプロパティー

7.6.2.20. ListPageCreateDropdown

権限チェックでラップされたドロップダウンを作成するためのコンポーネント。

例

```
const exampleList: React.FC<MyProps> = () => {
  const items = {
    SAVE: 'Save',
    DELETE: 'Delete',
  }
  return (
    <>
      <ListPageHeader title="Example Pod List Page"/>
      <ListPageCreateDropdown createAccessReview={access}>
        <ListPageCreateDropdown>Actions</ListPageCreateDropdown>
      </ListPageHeader>
      </>
    );
};
```

パラメータ名	説明
items	key: ドロップダウンコンポーネントに表示する項目の ReactNode のペア
onClick	ドロップダウン項目をクリックするためのコールバック関数
createAccessReview	(オプション) アクセスを決定するために使用される namespace と種類を持つオブジェクト
children	(オプション) ドロップダウントグルの子

7.6.2.21. ResourceLink

アイコンバッジを使用して特定のリソースタイプへのリンクを作成するコンポーネント。

例

```
<ResourceLink
  kind="Pod"
  name="testPod"
  title={metadata.uid}
/>
```

パラメータ名	説明
kind	(オプション) リソースの種類、つまり Pod、Deployment、Namespace
groupVersionKind	(オプション) グループ、バージョン、および種類を含むオブジェクト
className	(オプション) コンポーネントのクラススタイル
displayName	(オプション) コンポーネントの表示名。設定されている場合は、リソース名を上書きします。
inline	(オプション) アイコンバッジを作成し、子とインラインで名前を付けるためのフラグ
linkTo	(オプション) Link オブジェクトを作成するためのフラグ - デフォルトは true
name	(オプション) リソースの名前
namesapce	(オプション) リンク先の種類のリソースの特定の namespace
hidelcon	(オプション) アイコンバッジを非表示にするフラグ
title	(オプション) リンクオブジェクトのタイトル(非表示)
dataTest	(オプション) テスト用の識別子
onClick	(オプション) コンポーネントがクリックされたときのコールバック関数
truncate	(オプション) リンクが長すぎる場合に切り捨てるフラグ

7.6.2.22. Resourcelcon

特定のリソースタイプのアイコンバッジを作成するコンポーネント。

例

```
<Resourcelcon kind="Pod"/>
```

パラメータ名	説明
kind	(オプション) リソースの種類、つまり Pod、Deployment、Namespace
groupVersionKind	(オプション) グループ、バージョン、および種類を含むオブジェクト
className	(オプション) コンポーネントのクラススタイル

7.6.2.23. useK8sModel

指定された K8sGroupVersionKind の k8s モデルを redux から取得するフック。最初の項目が k8s モデル、2 番目の項目が **inFlight** ステータスの配列を返します。

例

```
const Component: React.FC = () => {
  const [model, inFlight] = useK8sModel({ group: 'app'; version: 'v1'; kind: 'Deployment' });
  return ...
}
```

パラメータ名	説明
groupVersionKind	k8s リソースのグループ、バージョン、種類。K8sGroupVersionKind が推奨されます。もしくは、グループ、バージョン、種類の参照 (例: group/version/kind (GVK) K8sResourceKindReference.) を渡すこともできますが、これは非推奨です。

7.6.2.24. useK8sModels

redux から現在のすべての k8s モデルを取得するフック。最初の項目が k8s モデルのリストで、2 番目の項目が **inFlight** ステータスの配列を返します。

例

```
const Component: React.FC = () => {
  const [models, inFlight] = UseK8sModels();
  return ...
}
```

}

7.6.2.25. useK8sWatchResource

ロード済みおよびエラーのステータスとともに k8s リソースを取得するフック。最初の項目がリソース、2番目の項目がロード済みステータス、3番目の項目がエラー状態(存在する場合)の配列を返します。

例

```
const Component: React.FC = () => {
  const watchRes = {
    ...
  }
  const [data, loaded, error] = useK8sWatchResource(watchRes)
  return ...
}
```

パラメーター名	説明
initResource	リソースを監視するために必要なオプション。

7.6.2.26. useK8sWatchResources

ロード済みおよびエラーのそれぞれのステータスとともに k8s リソースを取得するフック。キーが initResources で提供され、値が data、loaded、error の3つのプロパティーを持つマップを返します。

例

```
const Component: React.FC = () => {
  const watchResources = {
    'deployment': {...},
    'pod': {...}
    ...
  }
  const {deployment, pod} = useK8sWatchResources(watchResources)
  return ...
}
```

パラメーター名	説明
initResources	リソースはキーと値のペアとして監視する必要があります。ここで、キーはリソースに固有であり、値はそれぞれのリソースを監視するために必要なオプションです。

7.6.2.27. consoleFetch

コンソール固有のヘッダーを追加し、再試行とタイムアウトを可能にする **fetch** のカスタムラッパー。また、レスポンスステータスコードを検証し、適切なエラーを出力するか、必要に応じてユーザーをログアウトします。レスポンスに解決される promise を返します。

パラメーター名	説明
url	取得する URL
options	フェッチに渡すオプション
timeout	ミリ秒単位のタイムアウト

7.6.2.28. `consoleFetchJSON`

コンソール固有のヘッダーを追加し、再試行とタイムアウトを可能にする **fetch** のカスタムラッパー。また、レスポンスステータスコードを検証し、適切なエラーを出力するか、必要に応じてユーザーをログアウトします。レスポンスを JSON オブジェクトとして返します。内部で **consoleFetch** を使用します。JSON オブジェクトとしてレスポンスに解決される promise を返します。

パラメーター名	説明
url	取得する URL
method	使用する HTTP メソッドデフォルトは GET です。
options	フェッチに渡すオプション
timeout	ミリ秒単位のタイムアウト
cluster	リクエストを行うクラスターの名前。デフォルトは、ユーザーが選択したアクティブなクラスターです

7.6.2.29. `consoleFetchText`

コンソール固有のヘッダーを追加し、再試行とタイムアウトを可能にする **fetch** のカスタムラッパー。また、レスポンスステータスコードを検証し、適切なエラーを出力するか、必要に応じてユーザーをログアウトします。レスポンスをテキストとして返します。内部で **consoleFetch** を使用します。テキストとしてレスポンスに解決される promise を返します。

パラメーター名	説明
url	取得する URL
options	フェッチに渡すオプション
timeout	ミリ秒単位のタイムアウト

パラメータ名	説明
cluster	リクエストを行うクラスターの名前。デフォルトは、ユーザーが選択したアクティブなクラスターです

7.6.2.30. `getConsoleRequestHeaders`

redux の現在の状態を使用して、API 要求の偽装およびマルチクラスター関連のヘッダーを作成する関数。redux の状態に基づき、適切な偽装とクラスター要求ヘッダーを含むオブジェクトを返します。

パラメータ名	説明
targetCluster	指定された targetCluster で現在アクティブなクラスターをオーバーライドします

7.6.2.31. `k8sGetResource`

指定されたオプションに基づいて、クラスターからリソースを取得します。名前が指定されている場合は、1つのリソースが返されます。それ以外の場合は、モデルに一致するすべてのリソースが返されます。名前が指定されている場合、リソースを含む JSON オブジェクトとしてレスポンスに解決される promise を返します。それ以外の場合は、モデルに一致するすべてのリソースを返します。失敗した場合、promise は HTTP エラーレスポンスで拒否されます。

パラメータ名	説明
options	マップでキーと値のペアとして渡されるもの。
options.model	k8s モデル
options.name	リソースの名前。指定されていない場合は、モデルに一致するすべてのリソースが検索されます。
options.ns	検索対象の namespace。cluster-scoped リソースには指定しないでください。
options.path	指定されている場合はサブパスとして追加します
options.queryParams	URL に含めるクエリーパラメーター。
options.requestInit	使用する <code>fetch init</code> オブジェクト。これには、リクエストヘッダー、メソッド、リダイレクトなどを含めることができます。詳細は、 Interface RequestInit を参照してください。

7.6.2.32. `k8sCreateResource`

指定されたオプションに基づいて、クラスター内にリソースを作成します。作成されたリソースのレスポンスに解決される promise を返します。失敗した場合、promise は HTTP エラーレスポンスで拒否されます。

パラメーター名	説明
options	マップでキーと値のペアとして渡されるもの。
options.model	k8s モデル
options.data	作成されるリソースのペイロード
options.path	指定されている場合はサブパスとして追加します
options.queryParams	URL に含めるクエリーパラメーター。

7.6.2.33. `k8sUpdateResource`

指定されたオプションに基づいて、クラスター内のリソース全体を更新します。クライアントが既存のリソースを完全に置き換える必要がある場合、`k8sUpdate` を使用できます。または、`k8sPatch` を使用して部分的な更新を実行することもできます。更新されたリソースのレスポンスに解決される promise を返します。失敗した場合、promise は HTTP エラーレスポンスで拒否されます。

パラメーター名	説明
options	マップでキーと値のペアとして渡されます
options.model	k8s モデル
options.data	更新する k8s リソースのペイロード
options.ns	検索対象の namespace。cluster-scoped リソースには指定しないでください。
options.name	更新するリソース名。
options.path	指定されている場合はサブパスとして追加します
options.queryParams	URL に含めるクエリーパラメーター。

7.6.2.34. `k8sPatchResource`

指定されたオプションに基づいて、クラスター内の任意のリソースにパッチを適用します。クライアントが部分的な更新を実行する必要がある場合、`k8sPatch` を使用できます。または、`k8sUpdate` を使用して、既存のリソースを完全に置き換えることもできます。詳細は、[Data Tracker](#) を参照してください。パッチが適用されたリソースのレスポンスに解決される promise を返します。失敗した場合、promise は HTTP エラーレスポンスで拒否されます。

パラメーターネーム	説明
options	マップでキーと値のペアとして渡されるもの。
options.model	k8s モデル
options.resource	パッチを適用するリソース。
options.data	操作、パス、および値を含む既存のリソースにパッチを適用するデータのみ。
options.path	指定されている場合はサブパスとして追加します。
options.queryParams	URL に含めるクエリーパラメーター。

7.6.2.35. k8sDeleteResource

指定されたモデル、リソースに基づいて、クラスターからリソースを削除します。ガベージコレクションは **Foreground|Background** に基づいて機能し、指定されたモデルの `propagationPolicy` プロパティで設定するか、`json` で渡すことができます。種類が `Status` のレスポンスに解決される `promise` を返します。失敗した場合、`promise` は HTTP エラーレスポンスで拒否されます。

例

```
kind: 'DeleteOptions', apiVersion: 'v1', propagationPolicy
```

パラメーターネーム	説明
options	マップでキーと値のペアとして渡されるもの。
options.model	k8s モデル
options.resource	削除するリソース。
options.path	指定されている場合はサブパスとして追加します
options.queryParams	URL に含めるクエリーパラメーター。
options.requestInit	使用する <code>fetch init</code> オブジェクト。これには、リクエストヘッダー、メソッド、リダイレクトなどを含めることができます。詳細は、 Interface RequestInit を参照してください。
options.json	指定されている場合はリソースのガベージコレクションを明示的に制御できます。指定されていない場合は、モデルの <code>"propagationPolicy"</code> がデフォルトになります。

7.6.2.36. k8sListResource

指定されたオプションに基づいて、リソースをクラスター内の配列として一覧表示します。レスポンスに解決される promise を返します。

パラメーター名	説明
options	マップでキーと値のペアとして渡されるもの。
options.model	k8s モデル
options.queryParams	URL に含めるクエリーパラメーター。ラベルセレクターおよび "labelSelector" キーと併せて渡すことができます。
options.requestInit	使用する fetch init オブジェクト。これには、リクエストヘッダー、メソッド、リダイレクトなどを含めることができます。詳細は、 Interface RequestInit を参照してください。

7.6.2.37. k8sListResourceItems

k8sListResource と同じインターフェイスですが、サブ項目を返します。モデルの apiVersion、つまり **group/version** を返します。

7.6.2.38. getAPIVersionForModel

k8s モデルの apiVersion を提供します。

パラメーター名	説明
model	k8s モデル

7.6.2.39. getGroupVersionKindForResource

リソースのグループ、バージョン、および種類を提供します。指定されたリソースのグループ、バージョン、種類を返します。リソースに API グループがない場合、グループ "core" が返されます。リソースの apiVersion が無効な場合は、エラーが output されます。

パラメーター名	説明
resource	k8s リソース

7.6.2.40. getGroupVersionKindForModel

k8s モデルのグループ、バージョン、および種類を提供します。これは、提供されたモデルのグループ、バージョン、種類を返します。モデルに apiGroup がない場合は、グループ "core" が返されます。

パラメーターネーム	説明
model	k8s モデル

7.6.2.41. StatusPopupSection

ポップアップウィンドウでステータスを表示するコンポーネント。**console.dashboards/overview/health/resource** 拡張機能を構築するための便利なコンポーネント。

例

```
<StatusPopupSection
  firstColumn={
    <>
      <span>{title}</span>
      <span className="text-secondary">
        My Example Item
      </span>
    </>
  }
  secondColumn='Status'
>
```

パラメーターネーム	説明
firstColumn	ポップアップの最初の列の値
secondColumn	(オプション) ポップアップの 2 列目の値
children	(オプション) ポップアップの子

7.6.2.42. StatusPopupItem

ステータスポップアップで使用されるステータス要素。**StatusPopupSection** で使用されます。

例

```
<StatusPopupSection
  firstColumn='Example'
  secondColumn='Status'
>
  <StatusPopupItem icon={healthStateMapping[MCGMetrics.state]?.icon}>
    Complete
  </StatusPopupItem>
  <StatusPopupItem icon={healthStateMapping[RGWMetrics.state]?.icon}>
    Pending
  </StatusPopupItem>
</StatusPopupSection>
```

パラメーター名	説明
value	(オプション) 表示するテキスト値
icon	(オプション) 表示するアイコン
children	子要素

7.6.2.43. 概要

ダッシュボードのラッパコンポーネントを作成します。

例

```
<Overview>
  <OverviewGrid mainCards={mainCards} leftCards={leftCards} rightCards={rightCards} />
</Overview>
```

パラメーター名	説明
className	(オプション) div のスタイルクラス
children	(オプション) ダッシュボードの要素

7.6.2.44. OverviewGrid

ダッシュボードのカード要素のグリッドを作成します。 **Overview** 内で使用されます。

例

```
<Overview>
  <OverviewGrid mainCards={mainCards} leftCards={leftCards} rightCards={rightCards} />
</Overview>
```

パラメーター名	説明
mainCards	グリッド用カード
leftCards	(オプション) グリッドの左側のカード
rightCards	(オプション) グリッドの右側のカード

7.6.2.45. InventoryItem

インベントリーカード項目を作成します。

例

```

return (
<InventoryItem>
  <InventoryItemTitle>{title}</InventoryItemTitle>
  <InventoryItemBody error={loadError}>
    {loaded && <InventoryItemStatus count={workerNodes.length} icon={<MonitoringIcon />} />}
  </InventoryItemBody>
</InventoryItem>
)

```

パラメーター名	説明
children	項目内でレンダリングする要素

7.6.2.46. **InventoryItemTitle**

インベントリーカード項目のタイトルを作成します。**InventoryItem** 内で使用されます。

例

```

return (
<InventoryItem>
  <InventoryItemTitle>{title}</InventoryItemTitle>
  <InventoryItemBody error={loadError}>
    {loaded && <InventoryItemStatus count={workerNodes.length} icon={<MonitoringIcon />} />}
  </InventoryItemBody>
</InventoryItem>
)

```

パラメーター名	説明
children	タイトル内にレンダリングする要素

7.6.2.47. **InventoryItemBody**

インベントリーカードの本文を作成します。**InventoryCard** 内で使用され、**InventoryTitle** と使用できます。

例

```

return (
<InventoryItem>
  <InventoryItemTitle>{title}</InventoryItemTitle>
  <InventoryItemBody error={loadError}>
    {loaded && <InventoryItemStatus count={workerNodes.length} icon={<MonitoringIcon />} />}
  </InventoryItemBody>
</InventoryItem>
)

```

パラメータ名	説明
children	インベントリーカードまたはタイトル内でレンダリングする要素
error	div の要素

7.6.2.48. **InventoryItemStatus**

オプションのリンクアドレスを使用してインベントリーカードのカウントとアイコンを作成します。**InventoryItemBody** 内で使用されます。

例

```
return (
  <InventoryItem>
    <InventoryItemTitle>{title}</InventoryItemTitle>
    <InventoryItemBody error={loadError}>
      {loaded && <InventoryItemStatus count={workerNodes.length} icon={<MonitoringIcon />} />}
    </InventoryItemBody>
  </InventoryItem>
)
```

パラメータ名	説明
count	表示用カウント
icon	表示用アイコン
linkTo	(オプション) リンクアドレス

7.6.2.49. **InventoryItemLoading**

インベントリーカードのロード時にスケルトンコンテナーを作成します。**InventoryItem** および関連コンポーネントで使用されます。

例

```
if (loadError) {
  title = <Link to={workerNodesLink}>{t('Worker Nodes')}</Link>;
} else if (!loaded) {
  title = <><InventoryItemLoading /><Link to={workerNodesLink}>{t('Worker Nodes')}</Link></>;
}
return (
  <InventoryItem>
    <InventoryItemTitle>{title}</InventoryItemTitle>
  </InventoryItem>
)
```

7.6.2.50. useFlag

FLAGS redux 状態から指定された機能フラグを返すフック。要求された機能フラグまたは未定義の布尔値を返します。

パラメーター名	説明
flag	返す機能フラグ

7.6.2.51. CodeEditor

ホバーヘルプと補完機能を備えた基本的な遅延ロード Code エディター。

例

```
<React.Suspense fallback={<LoadingBox />}>
  <CodeEditor
    value={code}
    language="yaml"
  />
</React.Suspense>
```

パラメーター名	説明
value	レンダリングする yaml コードを表す文字列。
言語	エディターの言語を表す文字列。
options	Monaco エディターのオプション。 詳細は、 インターフェイス IStandAloneEditorConstructionOptions を参照してください。
minHeight	有効な CSS の高さの値における最小のエディターの高さ。
showShortcuts	エディターの上にショートカットを表示するための布尔値。
toolbarLinks	エディター上部のツールバーリンクセクションにレンダリングされる ReactNode の配列。
onChange	コード変更イベントのコールバック。
onSave	コマンド CTRL / CMD + S がトリガーされたときに呼び出されるコールバック。

パラメータ名	説明
ref	{ editor?: IStandaloneCodeEditor } への参照に反応します。editor プロパティーを使用すると、エディターを制御するすべてのメソッドにアクセスできます。詳細は、 インターフェイス IStandaloneCodeEditor を参照してください。

7.6.2.52. ResourceYAMLEditor

ホバーヘルプと補完機能を備えた Kubernetes リソース用の遅延ロード YAML エディター。このコンポーネントは YAMLEditor を使用し、その上にリソースの更新処理、アラート、保存、キャンセル、リロードボタン、アクセシビリティーなどの機能を追加します。onSave コールバックが提供されない限り、リソースの更新は自動的に処理されます。これは **React.Suspense** コンポーネントにラップする必要があります。

例

```
<React.Suspense fallback={<LoadingBox />}>
  <ResourceYAMLEditor
    initialResource={resource}
    header="Create resource"
    onSave={(content) => updateResource(content)}
  />
</React.Suspense>
```

パラメータ名	説明
initialResource	エディターによって表示されるリソースを表す YAML/オブジェクト。この prop は、最初のレンダリング中にのみ使用されます
header	YAML エディターの上にヘッダーを追加する
onSave	Save ボタンのコールバック。これを渡すと、エディターによってリソースに対して実行されるデフォルトの更新が上書きされます。

7.6.2.53. ResourceEventStream

特定のリソースに関連するイベントを表示するコンポーネント。

例

```
const [resource, loaded, loadError] = useK8sWatchResource(clusterResource);
return <ResourceEventStream resource={resource} />
```

パラメータ名	説明
resource	関連イベントを表示するオブジェクト。

7.6.2.54. usePrometheusPoll

単一のクエリーに対して Prometheus へのポーリングを設定します。クエリーレスポンス、レスポンスが完了したかどうかを示す布尔値フラグ、および要求中または要求の後処理中に発生したエラーを含むタプルを返します。

パラメータ名	説明
{PrometheusEndpoint} props.endpoint	PrometheusEndpoint (ラベル、クエリー、範囲、ルール、ターゲット) のいずれか
{string} [props.query]	(オプション) Prometheus クエリー文字列。空または未定義の場合、ポーリングは開始されません。
{number} [props.delay]	(オプション) ポーリング遅延間隔 (ミリ秒)
{number} [props.endTime]	(オプション) QUERY_RANGE エンドポイントの場合、クエリー範囲の終わり
{number} [props.samples]	(オプション) QUERY_RANGE エンドポイント用
{number} [options.timespan]	(オプション) QUERY_RANGE エンドポイント用
{string} [options.namespace]	(オプション) 追加する検索パラメーター
{string} [options.timeout]	(オプション) 追加する検索パラメーター

7.6.2.55. Timestamp

タイムスタンプをレンダリングするコンポーネント。タイムスタンプは、Timestamp コンポーネントの個々のインスタンス間で同期されます。指定されたタイムスタンプは、ユーザーロケールに従ってフォーマットされます。

パラメータ名	説明
timestamp	レンダリングするタイムスタンプ。形式は、ISO 8601 (Kubernetes で使用)、エポックタイムスタンプ、または日付のインスタンスであることが期待されます。
simple	アイコンとツールチップを省略したシンプルなバージョンのコンポーネントをレンダリングします。

パラメーターネーム	説明
omitSuffix	接尾辞を省略して日付をフォーマットします。
className	コンポーネントの追加のクラス名。

7.6.2.56. useOverlay

useOverlay フックは、Web コンソールのページ構造の外側の DOM にコンポーネントを直接挿入します。これにより、CSS を使用してコンポーネントを自由にスタイル設定し、配置できるようになります。たとえば、オーバーレイを UI の右上隅に浮かせる場合は、`style={{ position: 'absolute', right: '2rem', top: '2rem', zIndex: 999 }}` のようになります。**useOverlay** を複数回呼び出すことで、複数のオーバーレイを追加できます。**closeOverlay** 関数がオーバーレイコンポーネントに渡されます。これを呼び出すと、**useOverlay** で追加された可能性のある他のオーバーレイに影響を与えることなく、DOM からコンポーネントが削除されます。追加の props を **useOverlay** に渡すことができ、それらはオーバーレイコンポーネントに渡されます。

例

```
const OverlayComponent = ({ closeOverlay, heading }) => {
  return (
    <div style={{ position: 'absolute', right: '2rem', top: '2rem', zIndex: 999 }}>
      <h2>{heading}</h2>
      <Button onClick={closeOverlay}>Close</Button>
    </div>
  );
};

const ModalComponent = ({ body, closeOverlay, title }) => (
  <Modal isOpen onClose={closeOverlay}>
    <ModalHeader title={title} />
    <ModalBody>{body}</ModalBody>
  </Modal>
);

const AppPage: React.FC = () => {
  const launchOverlay = useOverlay();
  const onClickOverlay = () => {
    launchOverlay(OverlayComponent, { heading: 'Test overlay' });
  };
  const onClickModal = () => {
    launchOverlay(ModalComponent, { body: 'Test modal', title: 'Overlay modal' });
  };
  return (
    <Button onClick={onClickOverlay}>Launch an Overlay</Button>
    <Button onClick={onClickModal}>Launch a Modal</Button>
  )
}
```

7.6.2.57. ActionServiceProvider

console.action/provider 拡張タイプの他のプラグインからのコントリビューションを受け取ることを可能にするコンポーネント。

例

```
const context: ActionContext = { 'a-context-id': { dataFromDynamicPlugin } };

...
<ActionServiceProvider context={context}>
  {{ actions, options, loaded }} =>
    loaded && (
      <ActionMenu actions={actions} options={options} variant={ActionMenuVariant.DROPDOWN}>
    />
    )
  }
</ActionServiceProvider>
```

パラメーター名	説明
context	contextId とオプションのプラグインデータを含むオブジェクト

7.6.2.58. NamespaceBar

namespace のドロップダウンメニューが左端にある水平ツールバーをレンダリングするコンポーネント。追加のコンポーネントは子として渡すことができ、namespace ドロップダウンの右側にレンダリングされます。このコンポーネントは、ページの上部で使用するように設計されています。k8s リソースを含むページなど、ユーザーがアクティブな namespace を変更できる必要があるページで使用する必要があります。

例

```
const logNamespaceChange = (namespace) => console.log(`New namespace: ${namespace}`);

...
<NamespaceBar onNamespaceChange={logNamespaceChange}>
  <NamespaceBarApplicationSelector />
</NamespaceBar>
<Page>
  ...
</Page>
```

パラメーター名	説明
---------	----

パラメータ名	説明
onNamespaceChange	(オプション) namespace オプションが選択されたときに実行される関数。唯一の引数として、文字列の形式で新しい namespace を受け入れます。オプションが選択されると、アクティブな namespace が自動的に更新されますが、この関数を介して追加のロジックを適用できます。namespace が変更されると、URL の namespace パラメーターが以前の namespace から新しく選択された namespace に変更されます。
isDisabled	(オプション) true に設定されている場合、namespace のドロップダウンを無効にする布尔値フラグ。このオプションは namespace ドロップダウンにのみ適用され、子コンポーネントには影響しません。
children	(オプション) namespace ドロップダウンの右側にあるツールバー内にレンダリングされる追加の要素。

7.6.2.59. ErrorBoundaryFallbackPage

フルページの ErrorBoundaryFallbackPage コンポーネントを作成して、"Oh no!Something went wrong." というメッセージと、スタックトレースおよびその他の役立つデバッグ情報を表示します。これは、コンポーネントと組み合わせて使用されます。

例

```
//in ErrorBoundary component
return (
  if (this.state.hasError) {
    return <ErrorBoundaryFallbackPage errorMessage={errorString} componentStack={componentStackString}
      stack={stackTraceString} title={errorString}/>;
  }

  return this.props.children;
)
```

パラメータ名	説明
errorMessage	エラーメッセージのテキスト説明
componentStack	例外のコンポーネントトレース
stack	例外のスタックトレース
title	エラー境界ページのヘッダーとしてレンダリングするタイトル

7.6.2.60. QueryBrowser

Prometheus PromQL クエリーからの結果のグラフを、グラフと対話するためのコントロールとともにレンダリングするコンポーネント。

例

```
<QueryBrowser
  defaultTimespan={15 * 60 * 1000}
  namespace={namespace}
  pollInterval={30 * 1000}
  queries={[
    'process_resident_memory_bytes{job="console"}',
    'sum(irate(container_network_receive_bytes_total[6h:5m])) by (pod)',
  ]}
/>
```

パラメータ名	説明
customDataSource	(オプション) PromQL クエリーを処理する API エンドポイントのベース URL。指定した場合、これはデータをフェッチするためのデフォルト API の代わりに使用されます。
defaultSamples	(オプション) 各データ系列に対してプロットされるデータサンプルのデフォルトの数。データ系列が多い場合、QueryBrowser はここで指定した数よりも少ない数のデータサンプルを自動的に選択することができます。
defaultTimespan	(オプション) グラフのデフォルトのタイムスパン(ミリ秒単位) - デフォルトは 1,800,000 (30 分) です。
disabledSeries	(オプション) これらの正確なラベルと値のペアを持つデータシリーズを無効にします(表示しません)。
disableZoom	(オプション) グラフのズームコントロールを無効にするフラグ。
filterLabels	(オプション) 必要に応じて、返されたデータ系列をこれらのラベルと値のペアに一致するデータ系列のみにフィルタリングします。
fixedEndTime	(オプション) 現在の時刻までのデータを表示するのではなく、表示される時間範囲の終了時刻を設定します。
formatSeriesTitle	(オプション) 単一のデータ系列のタイトルとして使用する文字列を返す関数。

パラメータ名	説明
GraphLink	(オプション) 別のページへのリンクをレンダリングするためのコンポーネント (たとえば、このクエリーに関する詳細情報を取得する)。
hideControls	(オプション) グラフのタイムスパンなどを変更するためのグラフコントロールを非表示にするフラグ。
isStack	(オプション) 折れ線グラフの代わりに積み上げグラフを表示するフラグ。showStackedControl が設定されている場合でも、ユーザーは折れ線グラフに切り替えることができます。
namespace	(オプション) 指定した場合、この namespace のデータのみが返されます (この namespace ラベルを持つシリーズのみ)。
onZoom	(オプション) グラフがズームされたときに呼び出されるコールバック。
pollInterval	(オプション) 設定すると、最新のデータを表示するためにグラフが更新される頻度 (ミリ秒単位) が決まります。
queries	実行して結果をグラフに表示する PromQL クエリーの配列。
showLegend	(オプション) グラフの下に凡例を表示できるようにするフラグ。
showStackedControl	積み上げグラフモードと折れ線グラフモードを切り替えるためのグラフコントロールの表示を有効にするフラグ。
timespan	(オプション) グラフがカバーするタイムスパン (ミリ秒単位)。
units	(オプション) Y 軸およびツールチップに表示する単位。

7.6.2.61. useAnnotationsModal

Kubernetes リソースのアノテーションを編集するためのモーダルを起動するコールバックを提供する フック。

例

```
const PodAnnotationsButton = ({ pod }) => {
  const { t } = useTranslation();
  const launchAnnotationsModal = useAnnotationsModal<PodKind>(pod);
  return <button onClick={launchAnnotationsModal}>{t('Edit Pod Annotations')}

```

パラメータ名	説明
resource	K8sResourceCommon タイプのオブジェクトのアノテーションを編集するためのリソース。

戻り値

リソースのアノテーションを編集するためのモーダルを起動する関数。

7.6.2.62. useDeleteModal

リソースを削除するためのモーダルを起動するコールバックを提供するフック。

例

```
const DeletePodButton = ({ pod }) => {
  const { t } = useTranslation();
  const launchDeleteModal = useDeleteModal<PodKind>(pod);
  return <button onClick={launchDeleteModal}>{t('Delete Pod')}

```

パラメータ名	説明
resource	削除するリソース。
redirectTo	(オプション) リソースを削除した後にリダイレクトする場所。
message	(オプション) モーダルに表示するメッセージ。
btnText	(オプション) 削除ボタンに表示するテキスト。
deleteAllResources	(オプション) 同じ種類のリソースをすべて削除する機能。

戻り値

リソースを削除するためのモーダルを起動する関数。

7.6.2.63. useLabelsModel

Kubernetes リソースラベルを編集するためのモーダルを起動するコールバックを提供するフック。

例

```
const PodLabelsButton = ({ pod }) => {
  const { t } = useTranslation();
  const launchLabelsModal = useLabelsModal<PodKind>(pod);
  return <button onClick={launchLabelsModal}>{t('Edit Pod Labels')}</button>
}
```

パラメータ名	説明
resource	ラベルを編集するリソース (K8sResourceCommon タイプのオブジェクト)。

戻り値

リソースのラベルを編集するためのモーダルを起動する関数。

7.6.2.64. useActiveNamespace

現在アクティブな namespace と、アクティブな namespace を設定するためのコールバックを提供するフック。

例

```
const Component: React.FC = (props) => {
  const [activeNamespace, setActiveNamespace] = useActiveNamespace();
  return <select
    value={activeNamespace}
    onChange={(e) => setActiveNamespace(e.target.value)}
  >
    {
      // ...namespace options
    }
  </select>
}
```

戻り値

現在アクティブな namespace とセッターコールバックを含むタプル。

7.6.2.65. useUserSettings

ユーザー設定値と、ユーザー設定値を設定するためのコールバックを提供するフック。

例

```
const Component: React.FC = (props) => {
  const [state, setState, loaded] = useUserSettings(
    'devconsole.addPage.showDetails',
    true,
    true,
  );
  return loaded ? (

```

```

    <WrappedComponent {...props} userSettingState={state} setUserSettingState={setState} />
  ) : null;
};


```

戻り値

ユーザー設定値、セッターコールバック、およびロードされた布尔値を含むタプル。

7.6.2.66. useQuickStartContext

現在のクイックスタートコンテキスト値を提供するフック。これにより、プラグインがコンソールのクイックスタート機能と相互運用できるようになります。

例

```

const OpenQuickStartButton = ({ quickStartId }) => {
  const { setActiveQuickStart } = useQuickStartContext();
  const onClick = React.useCallback(() => {
    setActiveQuickStart(quickStartId);
  }, [quickStartId]);
  return <button onClick={onClick}>{t('Open Quick Start')}

```

戻り値

クイックスタートコンテキスト値オブジェクト。

7.6.2.67. PerspectiveContext

非推奨: 代わりに、指定された **usePerspectiveContext** を使用してください。パースペクティブコンテキストを作成します。

パラメータ名	説明
PerspectiveContextType	アクティブなパースペクティブとセッターを含むオブジェクト

7.6.2.68. useAccessReviewAllowed

非推奨: 代わりに **@console/dynamic-plugin-sdk** の **useAccessReview** を使用してください。指定されたリソースへのユーザーアクセスに関する使用可能なステータスを指定するフック。**isAllowed** ブール値を返します。

パラメータ名	説明
resourceAttributes	アクセスレビューのリソース属性
impersonate	権限借用の詳細

7.6.2.69. useSafetyFirst

非推奨: このフックはコンソールの機能とは関係ありません。指定されたコンポーネントがアンマウントされた場合に備えて、React 状態の安全な非同期設定を保証するフック。状態値とその set 関数のペアを含む配列を返します。

パラメーター名	説明
initialState	初期状態値

7.6.2.70. VirtualizedTable

非推奨: 代わりに PatternFly の [Data view](#) を使用してください。仮想化されたテーブルを作成するためのコンポーネント。

例

```
const MachineList: React.FC<MachineListProps> = (props) => {
  return (
    <VirtualizedTable<MachineKind>
      {...props}
      aria-label='Machines'
      columns={getMachineColumns}
      Row={getMachineTableRow}
    />
  );
}
```

パラメーター名	説明
data	テーブルのデータ
loaded	データがロードされたことを示すフラグ
loadError	データのロードで問題が発生した場合のエラーオブジェクト
列	列の設定
行	行の設定
unfilteredData	フィルターなしの元のデータ
NoDataEmptyMsg	(オプション) データのない空のメッセージコンポーネント
EmptyMsg	(オプション) 空のメッセージコンポーネント
scrollNode	(オプション) スクロールを処理する関数

パラメーターネーム	説明
label	(オプション) テーブルのラベル
ariaLabel	(オプション) aria ラベル
gridBreakPoint	応答性のためにグリッドを分割する方法のサイジング
onSelect	(オプション) テーブルの選択を処理する関数
rowData	(オプション) 行に固有のデータ

7.6.2.71. ListPageFilter

非推奨: 代わりに PatternFly の [Data view](#) を使用してください。リストページのフィルターを生成するコンポーネント。

例

```
// See implementation for more details on RowFilter and FilterValue types
const [staticData, filteredData, onFilterChange] = useListPageFilter(
  data,
  rowFilters,
  staticFilters,
);
// ListPageFilter updates filter state based on user interaction and resulting filtered data can be
rendered in an independent component.
return (
  <>
  <ListPageHeader .../>
  <ListPageBody>
    <ListPageFilter data={staticData} onFilterChange={onFilterChange} />
    <List data={filteredData} />
  </ListPageBody>
</>
)
```

パラメーターネーム	説明
data	データポイントの配列
loaded	データがロードされたことを示します
onFilterChange	フィルター更新時のコールバック関数
rowFilters	(オプション) 利用可能なフィルターオプションを定義する RowFilter 要素の配列

パラメータ名	説明
nameFilterPlaceholder	(オプション) 名前フィルターのプレースホルダー
labelFilterPlaceholder	(オプション) ラベルフィルターのプレースホルダー
hideLabelFilter	(オプション) 名前フィルターとラベルフィルターの両方ではなく、名前フィルターのみを表示します。
hideNameLabelFilter	(オプション) 名前フィルターとラベルフィルターの両方を非表示にします。
columnLayout	(オプション) 列レイアウトオブジェクト
hideColumnManagement	(オプション) 列管理を非表示にするフラグ

7.6.2.72. useListPageFilter

非推奨: 代わりに PatternFly の [Data view](#) を使用してください。ListPageFilter コンポーネントのフィルター状態を管理するフック。すべての静的フィルターによってフィルター処理されたデータ、すべての静的フィルターと行フィルターによってフィルター処理されたデータ、および rowFilters を更新するコールバックを含むタプルを返します。

例

```
// See implementation for more details on RowFilter and FilterValue types
const [staticData, filteredData, onFilterChange] = useListPageFilter(
  data,
  rowFilters,
  staticFilters,
);
// ListPageFilter updates filter state based on user interaction and resulting filtered data can be
// rendered in an independent component.
return (
  <>
  <ListPageHeader .../>
  <ListPageBody>
    <ListPageFilter data={staticData} onFilterChange={onFilterChange} />
    <List data={filteredData} />
  </ListPageBody>
  </>
)
)
```

パラメータ名	説明
data	データポイントの配列
rowFilters	(オプション) 利用可能なフィルターオプションを定義する RowFilter 要素の配列

パラメータ名	説明
staticFilters	(オプション) データに静的に適用される FilterValue 要素の配列

7.6.2.73. YAMLEditor

非推奨: 代わりに **CodeEditor** を使用してください。ホバーヘルプと補完機能を備えた基本的な遅延ロード YAML エディター。

例

```
<React.Suspense fallback={<LoadingBox />}>
  <YAMLEditor
    value={code}
  />
</React.Suspense>
```

パラメータ名	説明
value	レンダリングする yaml コードを表す文字列。
options	Monaco エディターのオプション。
minHeight	有効な CSS の高さの値における最小のエディターの高さ。
showShortcuts	エディターの上にショートカットを表示するためのブール値。
toolbarLinks	エディター上部のツールバーリンクセクションにレンダリングされる ReactNode の配列。
onChange	コード変更イベントのコールバック。
onSave	コマンド CTRL / CMD + S がトリガーされたときに呼び出されるコールバック。
ref	{ editor?: IStandaloneCodeEditor } への参照に反応します。editor プロパティを使用すると、エディターを制御するすべてのメソッドにアクセスできます。

7.6.2.74. useModal

非推奨: 代わりに `@console/dynamic-plugin-sdk` の `useOverlay` を使用してください。モーダルを起動するためのフック。

例

```
const AppPage: React.FC = () => {
  const launchModal = useModal();
  const onClick = () => launchModal(ModalComponent);
  return (
    <Button onClick={onClick}>Launch a Modal</Button>
  )
}
```

7.6.3. 動的プラグインのトラブルシューティング

プラグインのロードで問題が発生した場合は、このトラブルシューティングのヒントのリストを参照してください。

- 以下のコマンドを実行して、コンソールの Operator 設定でプラグインが有効になっており、プラグイン名が表示されていることを確認します。


```
$ oc get console.operator.openshift.io cluster -o jsonpath='{.spec.plugins}'
```

 - Overview ページのステータスカードで、有効になっているプラグインを確認します。プラグインが最近有効になった場合は、ブラウザーを更新する必要があります。
- 次の方法で、プラグインサービスが正常であることを確認します。
 - プラグイン Pod のステータスが実行中であり、コンテナーの準備が整っていることを確認します。
 - サービスラベルセレクターが Pod と一致し、ターゲットポートが正しいことを確認します。
 - コンソール Pod またはクラスター上の別の Pod のターミナルで、サービスから **plugin-manifest.json** をカールします。
- ConsolePlugin** リソース名 (**consolePlugin.name**) が **package.json** で使用されているプラグイン名と一致することを確認します。
- サービス名、namespace、ポート、およびパスが **ConsolePlugin** リソースで正しく宣言されていることを確認します。
- プラグインサービスが HTTPS とサービス提供証明書を使用していることを確認します。
- コンソール Pod ログで証明書または接続エラーを確認します。
- プラグインが依存する機能フラグが無効になっていないことを確認します。
- プラグインの **package.json** に一致しない **consolePlugin.dependencies** がないことを確認します。
 - これには、コンソールバージョンの依存関係または他のプラグインへの依存関係が含まれる場合があります。ブラウザーで JS コンソールをプラグインの名前でフィルタリングして、ログに記録されたメッセージを表示します。

- ナビゲーション拡張パースペクティブまたはセクション ID にタイプミスがないことを確認します。
 - プラグインはロードされている可能性がありますが、ID が正しくない場合、ナビゲーション項目が表示されません。URL を編集して、プラグインページに直接移動してみてください。
- コンソール Pod からプラグインサービスへのトラフィックをブロックしているネットワークポリシーがないことを確認します。
 - 必要に応じて、ネットワークポリシーを調整して、openshift-console namespace のコンソール Pod がサービスにリクエストを送信できるようにします。
- 開発者ツールブラウザーの **Console** タブで、ブラウザーにロードされる動的プラグインのリストを確認します。
 - **window.SERVER_FLAGS.consolePlugins** を評価して、コンソールフロントエンドの動的プラグインを確認します。

関連情報

- [サービス提供証明書について](#)

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。

8.1.1. 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。

8.1.2. 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 に関する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** オプションを使用すると、Web Terminal Operator の最新リリースをインストールできます。
 - **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** ページにリスト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ミナルアイコン () を確認します。

8.2. WEB 端末の設定

現在のセッションに対して(クラスター管理者の場合はすべてのユーザーセッションに対して)、Web ターミナルのタイムアウトとイメージの設定を設定できます。

8.2.1. セッションの Web 端末タイムアウトの設定

現在のセッションで、Web 端末のデフォルトタイムアウト期間を変更できます。

前提条件

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

手順



1. Web 端末アイコン()をクリックします。
2. オプション: 現在のセッションの Web 端末タイムアウトを設定します。
 - a. Timeout をクリックします。
 - b. 表示されるフィールドにタイムアウト値を入力します。
 - c. ドロップダウンリストから、タイムアウト間隔を **Seconds**、**Minutes**、**Hours**、または **Milliseconds** から選択します。
3. オプション: Web 端末で使用するカスタムイメージを選択します。
 - a. イメージをクリックします。
 - b. 表示されるフィールドに、使用するイメージの URL を入力します。
4. **Start** をクリックし、指定したタイムアウト設定を使用して端末インスタンスを開始します。

8.2.2. すべてのユーザーの Web ターミナルタイムアウトの設定

Web コンソールの **Administrator** パースペクティブを使用して、すべてのユーザーのデフォルトの Web ターミナルタイムアウト期間を設定できます。

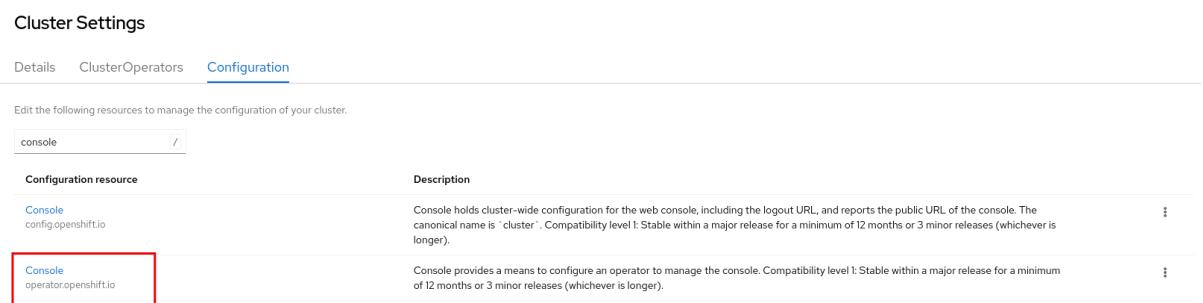
前提条件

- 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。
- Web Terminal Operator がインストールされている。

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。



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. Web Terminal タブをクリックすると、Web Terminal Configuration ページが開きます。

6. タイムアウトの値を設定します。ドロップダウンリストから、Seconds、Minutes、Hours、Milliseconds で時間の間隔を選択します。

7. Save をクリックします。

8.2.3. セッション用 Web 端末イメージの設定

現在のセッションで、Web 端末のデフォルトイメージを変更できます。

前提条件

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

手順



1. Web 端末アイコン () をクリックします。
2. Image をクリックして、Web 端末イメージの詳細設定オプションを表示します。
3. 使用するイメージの URL を入力します。
4. Start をクリックし、指定したイメージ設定を使用して端末インスタンスを開始します。

8.2.4. すべてのユーザーの Web 端末イメージを設定する

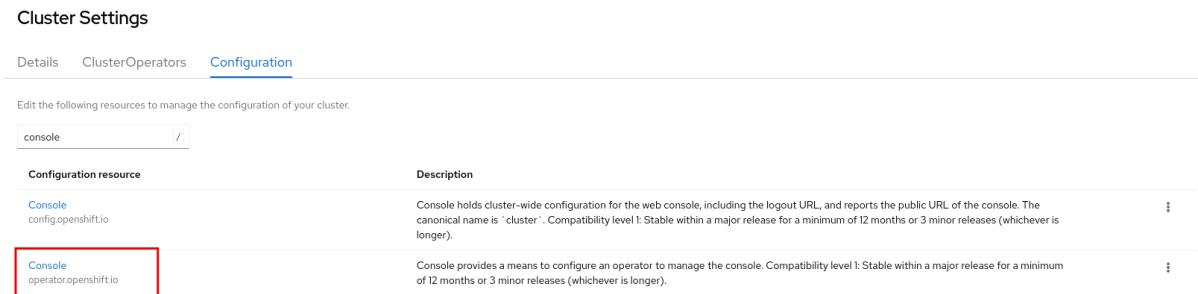
Web コンソールの Administrator パースペクティブを使用して、すべてのユーザーのデフォルトの Web ターミナルイメージを設定できます。

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。
- Web Terminal Operator がインストールされている。

手順

- Administrator パースペクティブで、Administration → Cluster Settings に移動します。
- Cluster Settings ページで、Configuration タブをクリックします。
- 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。



The screenshot shows the 'Cluster Settings' page with the 'Configuration' tab selected. A search bar at the top is set to 'console'. Below it, a table lists configuration resources. The first row, 'Console config.openshift.io', has its 'Description' column content: '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''. The second row, 'Console operator.openshift.io', has its 'Description' column content: '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)'. The second row is highlighted with a red box.

- 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。
- Web Terminal タブをクリックすると、Web Terminal Configuration ページが開きます。
- 使用するイメージの URL を入力します。
- Save をクリックします。

8.3. 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**、**kubectl**、**odo**、**kn**、**tkn**、**helm**、**subctl** など、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。

8.3.1. 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。Web 端末を初期化した後に、Web 端末で **oc**、**kubectl**、**odo**、**kn**、**tkn**、**helm**、**subctl** 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。ターミナルで実行したコマンドのリスト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. 現在のセッションでプロジェクトが選択されていない場合は、DevWorkspace CR を作成する必要があるプロジェクトを Project ドロップダウンリストから選択します。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

- 1つの DevWorkspace CR は、1ユーザーの Web 端末を定義します。この CR には、ユーザーの Web 端末ステータスおよびコンテナイメージコンポーネントに関する詳細が含まれています。
- 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。Web Terminal Operator は、DevWorkspace Operator を依存関係としてインストールします。

3. オプション: 現在のセッションの Web 端末タイムアウトを設定します。
 - a. Timeout をクリックします。
 - b. 表示されるフィールドにタイムアウト値を入力します。
 - c. ドロップダウンリストから、タイムアウト間隔を Seconds、Minutes、Hours、または Milliseconds から選択します。
4. オプション: Web 端末で使用するカスタムイメージを選択します。
 - a. イメージをクリックします。
 - b. 表示されるフィールドに、使用するイメージの URL を入力します。
5. 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。
6. + をクリックして、コンソールの Web 端末で複数のタブを開きます。

8.4. WEB 端末のトラブルシューティング

8.4.1. 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と、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起動するには、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。いずれのステップも失敗すると、Web 端末は起動に失敗し、context deadline exceeded error が発生するまで、端末パネルはロード状態になります。

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。

以下のサンプルは、**openshift-console** および **openshift-operators** namespace からの Ingress を許可する **NetworkPolicy** オブジェクトを示しています。

openshift-console namespace からの Ingress の許可

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-console
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: openshift-console
  podSelector: {}
  policyTypes:
  - Ingress
```

openshift-operators namespace からの Ingress の許可

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-operators
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: openshift-operators
  podSelector: {}
  policyTypes:
  - Ingress
```

8.5. 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。セキュリティー上の理由から、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. Web Terminal Operator と、Operator のインストール時に追加された関連カスタムリソース(CR) をアンインストールします。
2. Web Terminal Operator の依存関係として追加された DevWorkspace Operator と関連するカスタムリソースをアンインストールします。

8.5.1. 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

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにアクセスできる。
- **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。

8.5.2. 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 はスタンダード 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ーのコンテナー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ー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 OPENSHIFT 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ている。

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 ①
```

① `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** メニューでクリ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構成されます。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構成されます。

- **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。

- カードコピー
- はじめに
- タスクの手順
- モーダルとアプリケーション内メッセージング
- **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

- 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
```

- 以下を実行します。

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

- このドキュメント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。

- 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 ①
```

① **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。



Get started with Spring

⌚ 10 minutes

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

10.4.2.5. icon 要素

YAML ファイルでの icon 要素の表示

```
...
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
  displayName: Get started with Spring
  durationMinutes: 10
  icon: >- ①
    data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciGlkP$JMYXIlc8xliBkYXRhLW5hbWU9IkxheWVyIDEiIHZpZXdcB3g9IjAgMCAxMDI0IDEwMjQiPjxkZWZzPjxzcHlsZT4uY2xzLTf7ZmlsbDojMTUzZDNjO30uY2xzLTf7ZmlsbDojZDhkYTlkO30uY2xzLTf7ZmlsbDojNTkjMGE4O30uY2xzLTR7ZmlsbDojZmZmO30uY2xzLTf7ZmlsbDojM2Q5MTkxO308L3N0eWxIPjwvZGVmcz48dGl0bGU+c25vd2Ryb3BfaWNvbl9yZ2JfZGVmYXVsdDwvdGl0bGU+PHBhdGggY2xhc3M9ImNscy0xliBkPSJNMTAxMi42OSw1OTNjLTExLjEyLTM4LjA3LTmxLTcZLTU5LjIxLTEwMy44LTkuNS0xMS4zLTlZLjIxLTf4LjI5LTM5LjA2LTQ3Lj0QzgzMy41MywzNDEsNzQ1LjM3LDIzNC4xOCw2NzQsMTY4Ljk0Yy01LTUuMjYtMTAuMjYtMTAuMzEtMTUuNjUtMTUuMDdhMjQ2LjQ5LDI0Ni40OSwwLDAsMC0zNi41NS0yNi44LDE4Mi41LDE4Mi41LDAsMCwwLTlwljMtmTEuNzcsMjAxLjUzLDlwMS41MywwLDAsMC00My4xOS0xNUExNTUuMjQsMTU1LjI0LDAsMCwwLDUyOCw5NS4yYy02LjC2LS42OC0xMS43NC0uODEtMTQuMzktLjgxaDBsLTEuNjlsMC0xLjYyLDBhMTc3LjMsMTc3LjMsMCwwLDAAtMzEuNzcsMy4zNSwyMDguMjMsMjA4LjIzLDAAsMCwwLTU2LjEyLDE3LjU2LDE4MSwxODEsMCwwLDAAtMjAuMjcsMTEuNzUsMjQ3LjQzLDI0Ny40MywwLDAsMC0zNi41NywyNi44MUMzNjAuMjUsMTU4LjYyLDM1NSwxNjMuNjgsMzUwLDE2OWMtNzEuMzUsNjUuMjUtMTU5LjUsMTcyLTI0MC4zOSwyNzluMjhDOTMuNzMsNDYwLjg4LDgwLDQ3Ny44Nyw3MC41Miw0ODkuMTcsNDIuMzUsNTIwLDIyLjQzLDU1NC45LDExLjMxLDU5MywuNzIsNjI5LjlyLTEuNzMsNjY3LjY5LDQsNzA3LjMxLDE1LDc4Mi40OSw1NS43OCw4NTkuMTIsMTE4LjkzLDkyMy4wOWEyMiwyMiwwLDAsMCwxNS41OSw2LjUyaDEuODNsMS44Ny0uMzJjODEuMDYtMTMuOTEsMTEwLTc5LjU3LDE0My40OC0xNTUuNiwljKxLTguODgsNy45NS0xOC4wNSwxMi4yLTf3LjQzcTJuNDIsOC41NCwxMS4zOSwxNi4yM2MzMS44NSw0MC45MSw3NS4xMiw2NC42NywxMzluMzlsNzluNjNsMTguOCwyLjYyLDQuOTUtMTguMzNjMTMuMjYtNDkuMDcsMzUuMy05MC44NSw1MC42NC0xMTYuMTksMTUuMzQsMjUuMzQsMzcuMzgsNjcuMTIsNTAuNjQsMTE2LjE5bDUsMTguMzMsMTguOC0yLjYyYzU3LjltOCwxMDAuNDctMzEuNzlsMTMyLjMyLTcyLjYzcTYtNy42OCwxMS4zOS0xNi4yM2M0LjI1LDkuMzgsOC4yOSwxOC41NSwxMi4yLDI3LjQzLDMzLjQ5LDc2LDYyLjQyLDE0MS42OSwxNDMuNDgsMTU1LjZsMS44MS4zMWgxLjg5YtIyLDIyLDAsMCwwLDE1LjU5LTyUuNTJjNjMuMTUuNjQsMTAzaLjk1LTE0MC42LDExNC44OS0yMTUuNzhdMTAyNS43Myw2NjcuNjksMTAyMy4yOCw2MjkuMjlsMTAxMi42OSw1OTNali8+PHBhdGggY2xhc3M9ImNscy0yliBkPSJNMzY0LjE1LDE4NS4yM2MxNy44OS0xNi40LDM
```

```

OLjctMzAuMTUsNDkuNzctNDAuMTFhMjEyLDIxMiwwLDAsMSw2NS45My0yNS43M0ExOTgsMTk4LDA
sMCwxLDUxMiwxMTYuMjdhMTk2LjExLDE5Ni4xMSwwLDAsMSwzMiwzLjFjNC41LjxkxLDkuMzYsMi4wN
wxNC41MywzLjUyLDYwLjQxLDIwLjQ4LDg0LjkyLDkxLjA1LTQ3LjQ0LDI0OC4wNi0yOC43NSwzNC4x
Mi0xNDAuNywxOTQuODQtMTg0LjY2LDI2OC40MmE2MzAuODYsNjMwLjg2LDAsMCwwLTMzLjlyLD
U4LjMyQzI3Niw2NTUuMzQsMjY1LjQsNTk4LDI2NS40LDUyMC4yOSwyNjUuNCwzNDAuNjEsMzExLjY
5LDI0MC43NCwzNjQuMTUsMTg1LjIzWilvPjxwYXRoIGNsYXNzPSJjbHMtMyIgZD0iTTUyNy41NCwzO
DQuODNjODQuMDYtOTkuNywxMTYuMDYtMTc3LjI4LDk1LjlyLTIzMC43NCwxMS42Miw4LjY5LDI0LD
E5LjIsMzcuMDYsMzEuMTMsNTIuNDgsNTUuNSw5OC43OCwxNTUuMzgsOTguNzgsMzM1LjA3LDAs
NzcuNzEtMTAuNiwxMzUuMDUtMjcuNzcsMTc3LjRhNjI4LjczLDYyOC43MywwLDAsMC0zMy4yMy01OC
4zMmMtMzktNjUuMjYtMTMxLjQ1LTE5OS0xNzEuOTMtMjUyLjI3QzUyNi4zMywzODYuMjksNTI3LDM4
NS41Miw1MjcuNTQsMzg0LjgzWilvPjxwYXRoIGNsYXNzPSJjbHMtNCIgZD0iTTEzNC41OCw5MDguM
DdoLS4wNmEuMzkuMzksMCwwLDEtLjI3LS4xMWMtMTE5LjUyLTEyMS4wNy0xNTUtMjg3LjQtNDcuN
TQtNDA0LjU4LDM0LjYzLTQxLjE0LDEyMC0xNTEuNiwyMDIuNzUtmjQyLjE5LTMuMTMsNy02LjEyLDE
0LjI1LTguOTIsMjEuNjktMjQuMzQsNjQuNDUuMzYuNjcsMTQ0LjMyLTM2LjY3LDIzNy40MSwwLDU2LjU
zLDUuNTgsMTA2LDE2LjU5LDE0Ny4xNEEzMDcuNDksMzA3LjQ5LDAsMCwwLDI4MC45MSw3MjND
MjM3LDgxNi44OCwyMTYuOTMsODkzLjzLDEzNC41OCw5MDguMDdali8+PHBhdGggY2xhc3M9ImN
scy01liBkPSJNNTgzLjQzLDgxMy43OUM1NjAuMTgsNzI3LjcyLDUxMiw2NjQuMTUsNTEyLDY2NC4xN
XMtNDguMTcsNjMuNTctNzEuNDMsMTQ5LjY0Yy00OC40NS02Ljc0LTEwMC45MS0yNy41Mi0xMzUu
NjYtOTEuMTthhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Ny03MS41NGwuMjEtLjMyLjE5LS4zM2M
zOC02My42MywxMjYuNC0xOTEuMzcsMTY3LjEyLTI0NS42Niw0MC43MSw1NC4yOCwxMjkuMSwxO
DIsMTY3LjEyLDI0NS42NmwuMTkuMzMmuMjEuMzJhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Nyw
3MS41NEM2ODQuMzQsNzg2LjI3LDYzMS44OCw4MDcuMDUsNTgzLjQzLDgxMy43OVoiLz48cGF0a
CBjbGFycz0iY2xzLTQiIGQ9Ik04ODkuNzUsOTA4YS4zOS4zOSwwLDAsMS0uMjcuMTFoLS4wNkM4M
DcuMDcsODkzLjzLdc4Nyw4MTYuODgsNzQzLjA5LDcyM2EzMDcuNDksMzA3LjQ5LDAsMCwwLDIwL
jQ1LTU1LjU0YzExLTQxLjExLDE2LjU5LTkwLjYxLDE2LjU5LTE0Ny4xNCwwLTkzLjA4LTEyLjMzLTE3M
y0zNi42Ni0yMzcuNHEtNC4yMi0xMS4xNi04LjzLjIxDIuNzUsOTAuNTksMTY4LjEyLDIwMS4wNS
wyMDIuNzUsMjQyLjE5QzEwNDQuNzksNjIwLjU2LDEwMDkuMjcsNzg2Ljg5LDg4OS43NSw5MDhali8+
PC9zdmc+Cg==
...

```

1 base64 値として定義される icon。

Web コンソールでの icon 要素の表示

このアイコンは、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.

10.4.2.6. introduction 要素

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

  - 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、この中でタスクをリスト表示します。

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、この中でタスクをリスト表示します。

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



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,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdm
  cilHJvbGU9ImItZylgdmlld.
```

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リスト表示し、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

クリ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リックスタートを開始し、この中でタスクをリスト表示します。

- 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。
- クリックスタートの結果を示します。ユーザーは、クリ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(クリ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。
- 導入部分の後にクリックスタートのタスクをリスト表示し、各タスクのリスト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、クリックスタートを完了するために実行する必要のあるリスト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.

- 「In the top-right corner, click the icon」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:

The success message indicates a connection.

- 正しくない例:

The message with a green icon indicates a connection.

- 二人称を使用で統一します。

- 正しい例:

Set up your environment.

- 正しくない例:

Let's set up our environment.

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関する質問への 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は、[PatternFly の Brand voice and tone のガイドライン](#) を参照してください。
- 他の UX コンテンツのガイダンスは、[PatternFly の UX の作成ガイド \(writing style guide\)](#) のすべてのセクションを参照してください。

第11章 WEB コンソールのオプション機能と製品

製品を通して既存のワークフローやインテグレーションに機能を追加することで、OpenShift Container Platform Web コンソールをさらにカスタマイズできます。

11.1. OPERATOR を使用して OPENSHIFT CONTAINER PLATFORM WEB コンソールを拡張する

クラスター管理者は、OperatorHub を使用して OpenShift Container Platform Web コンソールのクラスターに Operator をインストールし、開発者向けにレイヤード製品の外部でカスタマイズを提供できます。たとえば、Web Terminal Operator を使用すると、クラスターと対話するための一般的な CLI ツールを使用してブラウザーで Web ターミナルを起動できます。

関連情報

- [OperatorHub について](#)
- [Web 端末のインストール](#)

11.2. WEB コンソールでの RED HAT OPENSHIFT LIGHTSPEED

Red Hat OpenShift Lightspeed は、OpenShift Container Platform 用の生成型人工知能を搭載した仮想アシスタントです。OpenShift Lightspeed 機能は、OpenShift Container Platform Web コンソールの自然言語インターフェイスを使用します。

この早期アクセスプログラムは、お客様がユーザーエクスペリエンス、特長、機能、遭遇した問題、他の機能についてフィードバックを提供できるため、リリースおよび一般公開時に OpenShift Lightspeed がお客様の要件にさらに適合できるようになります。

関連情報

- [OpenShift Lightspeed の概要](#)
- [OpenShift Lightspeed のインストール](#)

11.3. WEB コンソールの RED HAT OPENSHIFT PIPELINE

Red Hat OpenShift Pipelines は、Kubernetes リソースをベースとしたクラウドネイティブの継続的インテグレーションおよび継続的デリバリー (CI/CD) ソリューションです。OpenShift Container Platform Web コンソールの OperatorHub を使用して、Red Hat OpenShift Pipelines Operator をインストールします。Operator がインストールされると、パイプライン ページでパイプラインオブジェクトを作成および変更できます。

関連情報

- [Web コンソールでの Red Hat OpenShift Pipelines の使用](#)
- [Web コンソールのパイプライン実行に関する統計情報](#)

11.4. WEB コンソールの RED HAT OPENSHIFT SERVERLESS

Red Hat OpenShift Serverless を使用すると、開発者は OpenShift Container Platform 上でサーバーレスのイベント駆動型アプリケーションを作成およびデプロイできます。OpenShift Container Platform Web コンソール OperatorHub を使用して、OpenShift Serverless Operator をインストールできます。

関連情報

- [Web コンソールから OpenShift Serverless Operator をインストールする](#)

11.5. OPENSHIFT CONTAINER PLATFORM WEB コンソールの RED HAT DEVELOPER HUB

Red Hat Developer Hub は、合理化された開発環境を体験するために使用できるプラットフォームです。Red Hat Developer Hub は一元化されたソフトウェアカタログを採用しており、マイクロサービスとインフラストラクチャーの効率向上を実現します。これにより、製品チームは妥協することなく高品質のコードを提供できるようになります。クイックスタートでは、開発者ハブのインストール方法の詳細を確認できます。

11.5.1. OpenShift Container Platform Web コンソールを使用した Red Hat Developer Hub のインストール

Web コンソールは、Red Hat Developer Hub Operator のインストール方法に関するクイックスタートを提供します。

前提条件

- **cluster-admin** 特権で OpenShift Container Platform Web コンソールにログインしている。

手順

1. Overview ページで、Getting started resources タイルの **Install Red Hat Developer Hub (RHDH) with an Operator** をクリックします。
2. Operator で Red Hat Developer Hub をインストールする手順とともに、クイックスタートペインが表示されます。Operator のインストール、Red Hat Developer Hub インスタンスの作成、およびインスタンスを **OpenShift Console Application** メニューに追加する方法は、クイックスタートに従ってください。

検証

1. 表示される **Application launcher** のリンクをクリックして、**Application** タブが使用可能であることを確認できます。
2. Janus IDP インスタンスを開くことができることを確認します。

関連情報

- [Red Hat Developer Hub の製品ドキュメント](#)