OpenShift Container Platform 4.8

Web コンソール

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

Last Updated: 2021-11-21

OpenShift Container Platform 4.8 Web コンソール

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

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

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

目次

第1章 WEB コンソールへのアクセス

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ースです。開発者は Web コンソールを使用してプロジェクトのコンテンツを視覚的に把握し、参照し、管理することができます。

1.1. 前提条件

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

1.2. WEB コンソールの理解および WEB コンソールへのアクセス

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

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。

第2章 OPENSHIFT CONTAINER PLATFORM DASHBOARD を使用したクラスター情報の取得

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

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

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

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

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

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

第3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、コンソールを無効にしたりすることができます。

3.1. 前提条件

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

3.2. WEB コンソールの設定

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

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

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

以下の例は、コンソールのリソース定義のサンプルを示しています。

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

(1) ユーザーが Web コンソールからログアウトする際に読み込むページの 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。

(2) Web コンソール URL。これをカスタム値に更新するには、「**Web コンソール URL のカスタマイズ**」を参照してください。

第4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。

4.1. カスタムロゴおよび製品名の追加

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

前提条件

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

手順

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

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: console-custom-logo
  namespace: openshift-config
data:
  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。

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

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

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

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

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

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

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

コンソール Operator 設定を確認するには、以下を実行します。

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

4.2. WEB コンソールでのカスタムリンクの作成

前提条件

- 管理者の権限があること。

手順

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

```
apiVersion: console.openshift.io/v1
kind: ConsoleLink
metadata:
  name: example
spec:
  href: 'https://www.example.com'
  location: HelpMenu 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. **保存」**をクリックすると、変更内容が適用されます。

4.3. 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。

4.3.1. コンソールルートのカスタマイズ

クラスタ**Ingress**構成の**spec.componentRoutes**フィールドにカスタムホスト名とTLS証明書を設定することで、コンソールルートをカスタマイズすることができます。

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サフィックスのドメインが、クラスターのドメインサフィックスと一致しない場合に必要です。サフィックスが一致する場合には、シークレットはオプションです。

ヒント

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

手順

1. クラスター **Ingress** 設定を編集します。

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

2. カスタムホスト名を設定し、オプションでサービング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. カスタムホスト名です。
2. TLS 証明hそ (**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サフィックスのドメインが、クラスターのドメインサフィックスと一致しない場合に必要です。サフィックスが一致する場合には、シークレットはオプションです。

3. 変更を適用するためにファイルを保存します。

4.3.2. ダウンロードルートのカスタマイズ

クラスタ**Ingress**構成の**spec.componentRoutes**フィールドにカスタムホスト名とTLS証明書を設定することで、ダウンロードルートをカスタマイズすることができます。

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サフィックスのドメインが、クラスターのドメインサフィックスと一致しない場合に必要です。サフィックスが一致する場合には、シークレットはオプションです。

ヒント

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

手順

1. クラスター **Ingress** 設定を編集します。

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

2. カスタムホスト名を設定し、オプションでサービング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. カスタムホスト名です。
2. TLS 証明hそ (**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サフィックスのドメインが、クラスターのドメインサフィックスと一致しない場合に必要です。サフィックスが一致する場合には、シークレットはオプションです。

3. 変更を適用するためにファイルを保存します。

4.4. 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ープロバイダー (IDP) に限定されます。LDAP や HTPasswd などのダイレクトパスワード認証を使用する IDP にはこれによる影響がありません。

前提条件

- 管理者の権限があること。

手順

1. 以下のコマンドを実行して、変更可能なテンプレートを作成します。

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

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

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

2. シークレットを作成します。

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

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

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

3. 以下を実行します。

```
$ oc edit oauths cluster
```

4. 仕様を更新します。

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

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

4.5. 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

- 管理者の権限があること。

手順

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

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

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

4.6. カスタム通知バナーの作成

前提条件

- 管理者の権限があること。

手順

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

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

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

4. **Create**」をクリックすると、変更内容が適用されます。

4.7. CLI ダウンロードのカスタマイズ

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

前提条件

- 管理者の権限があること。

手順

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

```
apiVersion: console.openshift.io/v1
kind: ConsoleCLIDownload
metadata:
  name: example-cli-download-links-for-foo
spec:
  description: |
    This is an example of download links for foo
  displayName: example-foo
  links:
  - href: 'https://www.example.com/public/foo.tar'
    text: foo for linux
  - href: 'https://www.example.com/public/foo.mac.zip'
    text: foo for mac
  - href: 'https://www.example.com/public/foo.win.zip'
    text: foo for windows
```

4. **Save** ボタンをクリックします。

4.8. 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。

第5章 WEB コンソールの DEVELOPER パースペクティブ

OpenShift Container Platform Web コンソールは、**Administrator** パースペクティブと **Developer** パースペクティブという 2 つのパースペクティブを提供します。

注記

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

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

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

5.1. 前提条件

Developer パースペクティブにアクセスするために、Web コンソールにログインしていること。

5.2. DEVELOPER パースペクティブへのアクセス

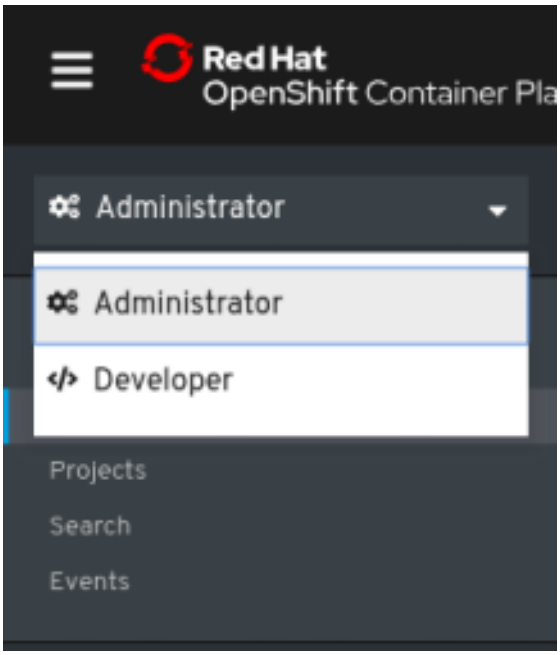
OpenShift Container Platform Web コンソールの **Developer** パースペクティブは、開発者のユースケースに固有のワークフローを提供します。

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

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。OpenShift Container Platform Web コンソールのデフォルトビューは **Administrator** パースペクティブです。
2. パースペクティブスイッチャーを使用して、**Developer** パースペクティブに切り替えます。**Topology** ビューがクラスター内のすべてのプロジェクトの一覧と共に表示されます。

図5.1 Developer パースペクティブ



3. 一覧から既存プロジェクトを選択するか、または **Project** ドロップダウンリストを使用して新規プロジェクトを作成します。

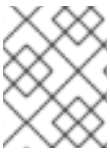
プロジェクト内にワークロードやアプリケーションがない場合、**Topology** ビューにはアプリケーションを作成するための利用可能なオプションが表示されます。既存のワークロードがある場合、**Topology** ビューはワークロードノードをグラフィカルに表示します。

追加リソース

- **開発者**視点でのOpenShift Container Platform上でのアプリケーションの作成とデプロイ
- プロジェクトでのアプリケーションの表示

第6章 WEB コンソールの WEB 端末について

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。Web 端末を使用するには、まず Web 端末 Operator をインストールする必要があります。



注記

クラスター管理者は、OpenShift Container Platform 4.7 以降の Web 端末にアクセスできます。

この端末のインスタンスは、**oc**、**kubectl**、**odo**、**kn**、**tkn**、**helm**、**kubens**、および **kubectx** などのクラスターと対話するための一般的な CLI ツールと共に事前にインストールされます。また、これには作業しているプロジェクトのコンテキストが含まれ、ユーザーの認証情報を使用してユーザーのログインを自動的に行います。

重要

Web 端末はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

6.1. WEB 端末のインストール

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

前提条件

- **cluster-admin** パーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスする。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → OperatorHub** に移動します。
2. **Filter by keyword** ボックスを使用してカタログで **Web Terminal** Operator を検索し、**Web Terminal** タイルをクリックします。
3. **Web Terminal** ページで Operator についての簡単な説明を確認してから、**Install** をクリックします。
4. **Install Operator** ページで、すべてのフィールドのデフォルト値を保持します。

- **Update Channel** メニューの **fast** オプションは、Web 端末 Operator の最新リリースのインストールを可能にします。
- **Installation Mode** メニューの **All namespaces on the cluster**オプションにより、Operator にクラスターのすべての namespace を監視され、Operator をこれらの namespace で利用可能にすることができます。
- **Installed Namespace** メニューの **openshift-operators** オプションは、Operator をデフォルトの **openshift-operators** namespace にインストールします。
- **Approval Strategy** メニューの **Automatic** オプションにより、Operator への今後のアップグレードは Operator Lifecycle Manager によって自動的に処理されます。

5. **Install** をクリックします。
6. **Installed Operators** ページで、**View Operator** をクリックし、Operator が **Installed Operators** ページに一覧表示されていることを確認します。

注記

OpenShift Container Platform 4.8 よりも前のバージョンでは、Web 端末 Operator はそのすべての機能を単一 Operator にバンドルしていました。OpenShift Container Platform 4.8 では、Web 端末 Operator は依存関係として DevWorkspace Operator をインストールし、同じ機能を提供します。

7. Operator のインストール後に、ページを更新し、コンソールの右上にあるコマンドラインターミナルアイコンを確認します。

6.2. WEB 端末の使用

Web 端末 Operator のインストール後に、以下のように Web 端末を使用できます。

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

注記

- **DevWorkspace** CR は存在しない場合にのみ作成されます。
- **openshift-terminal** プロジェクトは、クラスター管理者に使用されるデフォルトのプロジェクトです。別のプロジェクトを選択するオプションはありません。

3. **Start** をクリックし、選択したプロジェクトを使用して Web 端末を初期化します。

Web 端末を初期化した後に、Web 端末で **oc**、**kubectl**、**odo**、**kn**、**tkn**、**helm**、**kubens** および **kubectx** などの事前インストールされた CLI ツールを使用できます。

6.3. WEB 端末のアンインストール

Web 端末のアンインストールは 2 つの手順で実行されます。

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

Web 端末 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。これらのコンポーネントは、セキュリティー上の目的で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 のアンインストール時に端末がアイドル状態にならないようにしてクラスターリソースを保存することもできます。

前提条件

- **cluster-admin** パーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスする。

6.3.1. Web 端末 Operator およびこれをサポートするカスタムリソースの削除

コンソールおよび CLI を使用して、Web 端末 Operator のインストール時に作成された既存の Web 端末および CR を削除します。

注記

OpenShift Container Platform 4.8 よりも前のバージョンでは、Web 端末 Operator は Web 端末機能を提供するために異なる CRD を使用していました。Web 端末 Operator のバージョン 1.2.1 以前をアンインストールするには、OpenShift Container Platform 4.7 のドキュメントを参照してください。

手順

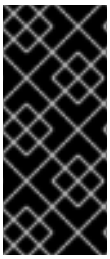
1. Web コンソールを使用して Web 端末 Operator をアンインストールします。
   a. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
   b. フィルター一覧をスクロールするか、または **Filter by name** ボックスにキーワードを入力して **Web 端末** Operator を見つけます。
   c. Web 端末 Operator の Options メニュー  をクリックし、**Uninstall Operator** を選択します。
   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。
2. 以下のコマンドを実行して、Web 端末 Operator 用に作成されたすべての **DevWorkspace** CR が削除されていることを確認します。

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

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

6.3.2. DevWorkspace Operator 依存関係の削除

CLI を使用して、Web 端末 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) および追加のリソースを削除します。



重要

DevWorkspace Operator はスタンドアロン Operator として機能し、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 (たとえば、CodeReady Workspaces Operator はこれに依存する場合があります)。DevWorkspace Operator が不要であることが確実な場合にのみ、以下の手順に従ってください。

手順

1. 以下のコマンドを実行して、すべての **DevWorkspace** CR がデプロイメントなどの関連する Kubernetes オブジェクトと共に削除されるようにします。

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

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

警告

この手順が完了しない場合、ファイナライザーは Operator を簡単に、かつ完全にアンインストールすることができないようにします。

2. CRD を削除するには、以下のコマンドを実行します。

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

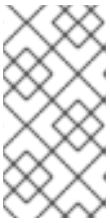
```
$ oc delete customresourcedefinitions.apiextensions.k8s.io
devworkspaces.workspace.devfile.io
```

3. **DevWorkspace-Webhook-Server** のデプロイメント、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すると、**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することはできません。Webhook を削除すると、**oc exec** コマンドを再び使用できるようになります。

4. 以下のコマンドを実行して依然として存在しているサービス、シークレット、および設定マップを削除します。

```
$ oc delete all --selector app.kubernetes.io/part-of=devworkspace-operator,app.kubernetes.io/name=devworkspace-webhook-server
```

```
$ oc delete serviceaccounts devworkspace-webhook-server -n openshift-operators
```

```
$ oc delete configmap devworkspace-controller -n openshift-operators
```

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

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

5. Web コンソールを使用して 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. フィルター一覧をスクロールするか、または **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. DevWorkspace Operator の Options 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7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。

7.1. 前提条件

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

7.2. 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8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。

8.1. 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。以下は例になります。

- 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

8.2. 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。

8.3. 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。

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

8.4. クイックスタートの継続

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

前提条件

- クラスター管理者の権限があること。

手順

1. 新規のクイックスタートを作成するには、以下を実行します。

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

2. 以下を実行します。

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

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

8.4.1. クイックスタート API ドキュメントの表示

手順

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

```
$ oc explain consolequickstarts
```

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

8.4.2. 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。

8.4.2.1. conclusion 要素

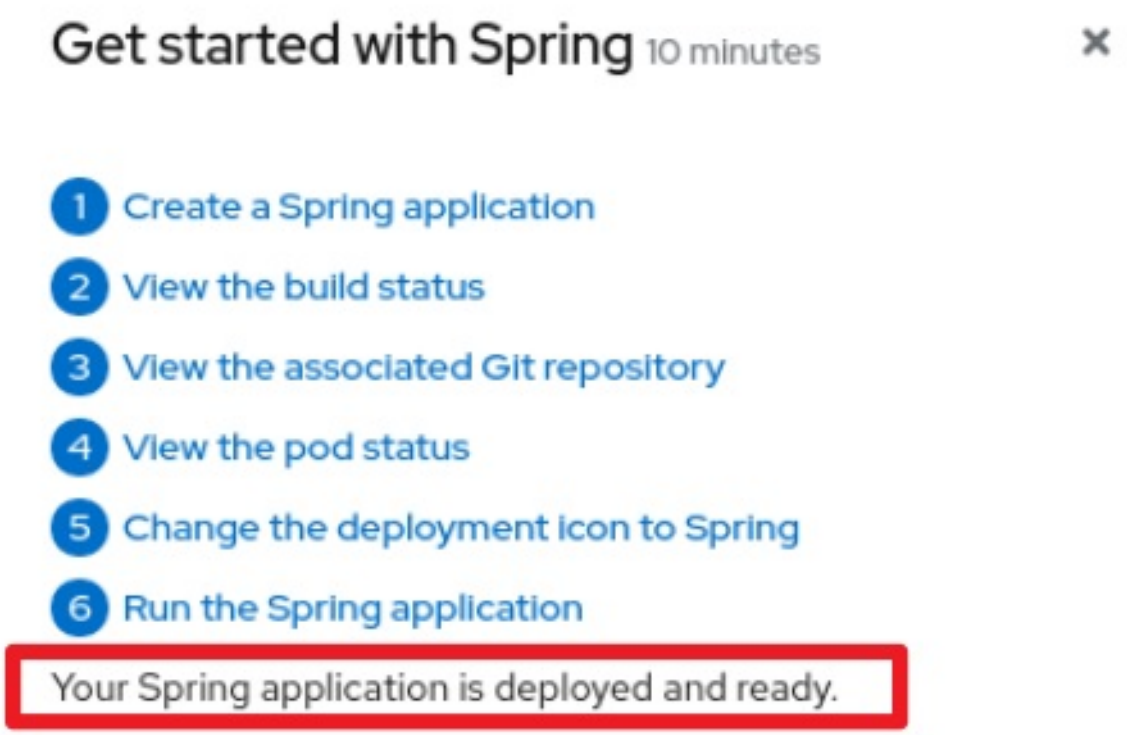
YAML ファイルの conclusion 要素の表示

```
...
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。



8.4.2.2. description 要素

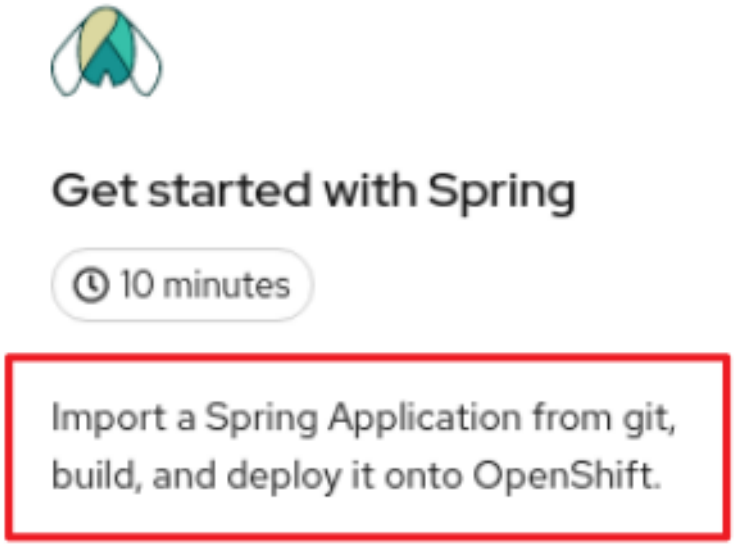
YAML ファイルでの description 要素の表示

```
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。



8.4.2.3. displayName 要素

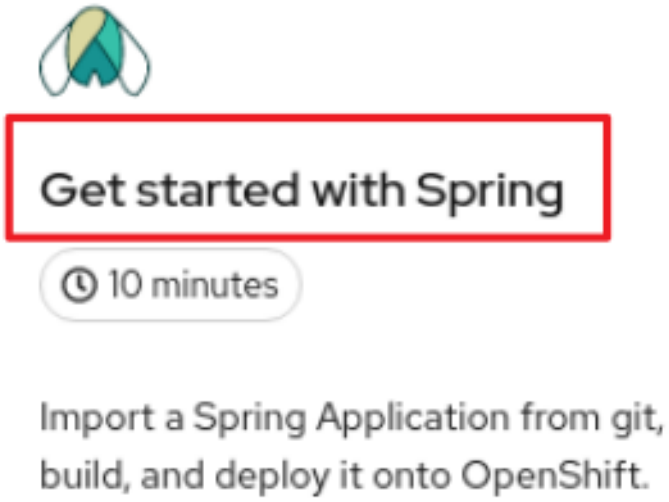
YAML ファイルの displayName 要素の表示

```
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。



8.4.2.4. durationMinutes 要素

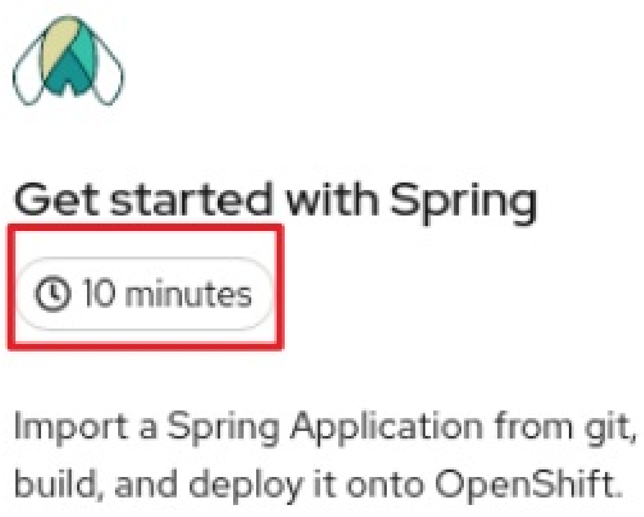
YAML ファイルでの durationMinutes 要素の表示

```
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。



8.4.2.5. icon 要素

YAML ファイルでの icon 要素の表示

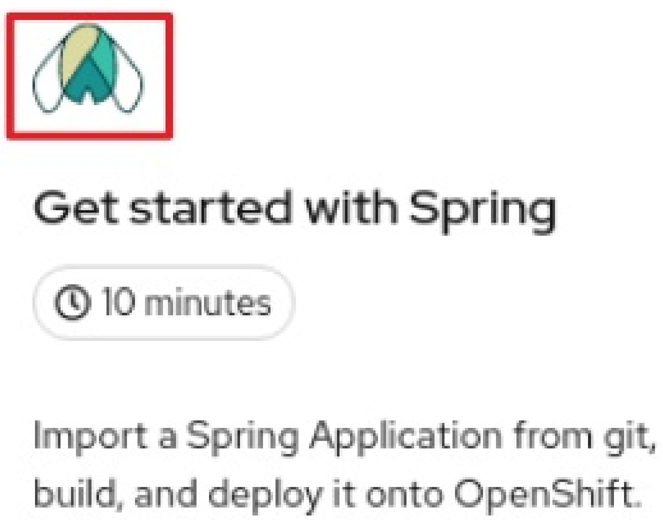
```
...
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
  displayName: Get started with Spring
  durationMinutes: 10
  icon: >-  (1)


JMYXllcl8xIiBkYXRhLW5hbWU9IkxheWVyIDEiIHZpZXdCb3g9IjAgMCAxMDI0IDEwMjQiPjxkZWZzPjxz
HlsZT4uY2xzLTF7ZmlsbDojMTUzZDNjO30uY2xzLTJ7ZmlsbDojZDhkYTlkO30uY2xzLTN7ZmlsbDojNT
hjMGE4O30uY2xzLTR7ZmlsbDojZmZmO30uY2xzLTV7ZmlsbDojM2Q5MTkxO308L3N0eWxlPjwvZGV
mcz48dGl0bGU+c25vd2Ryb3BfaWNvbl9yZ2JfZGVmYXVsdDwvdGl0bGU+PHBhdGggY2xhc3M9ImNsc
y0xIiBkPSJNMTAxMi42OSw1OTNjLTExLjEyLTM4LjA3LTMxLTczLTU5LjIxLTEwMy44LTkuNS0xMS4zL
TIzLjIxLTI4LjI5LTM5LjA2LTQ3Ljk0QzgzMy41MywzNDEsNzQ1LjM3LDIzNC4xOCw2NzQsMTY4Ljk0Yy
01LTUuMjYtMTAuMjYtMTAuMzEtMTUuNjUtMTUuMDdhMjQ2LjQ5LDI0Ni40OSwwLDAsMC0zNi41NS
0yNi44LDE4Mi41LDE4Mi41LDAsMCwwLTIwLjMtMTEuNzcsMjAxLjUzLDIwMS41MywwLDAsMC00My4
xOS0xNUExNTUuMjQsMTU1LjI0LDAsMCwwLDUyOCw5NS4yYy02Ljc2LS42OC0xMS43NC0uODEtM
TQuMzktLjgxaDBsLTEuNjIsMC0xLjYyLDBhMTc3LjMsMTc3LjMsMCwwLDAtMzEuNzcsMy4zNSwyMDg
uMjMsMjA4LjIzLDAsMCwwLTU2LjEyLDE3LjU2LDE4MSwxODEsMCwwLDAtMjAuMjcsMTEuNzUsMjQ
3LjQzLDI0Ny40MywwLDAsMC0zNi41NywyNi44MUMzNjAuMjUsMTU4LjYyLDM1NSwxNjMuNjgsMzUw
LDE2OWMtNzEuMzUsNjUuMjUtMTU5LjUsMTcyLTI0MC4zOSwyNzIuMjhDOTMuNzMsNDYwLjg4LDg
wLDQ3Ny44Nyw3MC41Miw0ODkuMTcsNDIuMzUsNTIwLDIyLjQzLDU1NC45LDExLjMxLDU5MywuNz
IsNjI5LjIyLTEuNzMsNjY3LjY5LDQsNzA3LjMxLDE1LDc4Mi40OSw1NS43OCw4NTkuMTIsMTE4LjkzLDk
kyMy4wOWEyMiwyMiwwLDAsMCwxNS41OSw2LjUyaDEuODNsMS44Ny0uMzJjODEuMDYtMTMuOT
EsMTEwLTc5LjU3LDE0My40OC0xNTUuNiwzLjkxLTguODgsNy45NS0xOC4wNSwxMi4yLTI3LjQzcTU
uNDIsOC41NCwxMS4zOSwxNi4yM2MzMS44NSw0MC45MSw3NS4xMiw2NC42NywxMzIuMzIsNzIuNj
NsMTguOCwyLjYyLDQuOTUtMTguMzNjMTMuMjYtNDkuMDcsMzUuMy05MC44NSw1MC42NC0xMT
YuMTksMTUuMzQsMjUuMzQsMzcuMzgsNjcuMTIsNTAuNjQsMTE2LjE5bDUsMTguMzMsMTguOC0yL
jYyYzU3LjItOCwxMDAuNDctMzEuNzIsMTMyLjMyLTcyLjYzLDQtNS4xMyw3LjgxLDEwLjUyLDExLjI1L
DkuMzgsOC4yOSwxOC41NSwxMi4yLDI3LjQzLDMzLjQ5LDc2LDYyLjQyLDE0MS42OSwxNDMuNDgs
MTU1LjZsMS44Ny4zMWgxLjg5YTIyLDIyLDAsMCwwLDE1LjU5LTYuNTJjNjMuMTUtNjQsMTAzLjk1LT
E0MC42LDExNC44OS0yMTUuNzhDMTAyNS43OCw2NjAuNjksMTAyMy4yOCw2MjkuMjIsMTAxMi42O
Sw1OTNaIi8+PHBhdGggY2xhc3M9ImNscy0yIiBkPSJNMzY0LjE1LDE4NS4yM2MxNy44OS0xNi40LDM
```

```
0LjctMzAuMTUsNDkuNzctNDAuMTFhMjEyLDIxMiwwLDAsMSw2NS45My0yNS43M0ExOTgsMTk4LDA
sMCwxLDUxMiwxMTYuMjdhMTk2LjExLDE5Ni4xMSwwLDAsMSwzMiwzLjFjNC41LjkxLDkuMzYsMi4wN
wxNC41MywzLjUyLDYwLjQxLDIwLjQ4LDg0LjkyLDkxLjA1LTQ3LjQ0LDI0OC4wNi0yOC43NSwzNC4x
Mi0xNDAuNywxOTQuODQtMTg0LjY2LDI2OC40MmE2MzAuODYsNjMwLjg2LDAsMCwwLTMzLjIyLD
U4LjMyQzI3Niw2NTUuMzQsMjY1LjQsNTk4LDI2NS40LDUyMC4yOSwyNjUuNCwzNDAuNjEsMzExLjY
5LDI0MC43NCwzNjQuMTUsMTg1LjIzWiIvPjxwYXRoIGNsYXNzPSJjbHMtMyIgZD0iTTUyNy41NCwzO
DQuODNjODQuMDYtOTkuNywxMTYuMDYtMTc3LjI4LDk1LjIyLTIzMC43NCwxMS42Miw4LjY5LDI0LD
E5LjIsMzcuMDYsMzEuMTMsNTIuNDgsNTUuNSw5OC43OCwxNTUuMzgsOTguNzgsMzM1LjA3LDAs
NzcuNzEtMTAuNiwxMzUuMDUtMjcuNzcsMTc3LjRhNjI4LjczLDYyOC43MywwLDAsMC0zMy4yMy01OC
4zMmMtMzktNjUuMjYtMTMxLjQ1LTE5OS0xNzEuOTMtMjUyLjI3QzUyNi4zMywzODYuMjksNTI3LDM4
NS41Miw1MjcuNTQsMzg0LjgzWiIvPjxwYXRoIGNsYXNzPSJjbHMtNCIgZD0iTTEzNC41OCw5MDguM
DdoLS4wNmEuMzkuMzksMCwwLDEtLjI3LS4xMWMtMTE5LjUyLTEyMS4wNy0xNTUtMjg3LjQtNDcuN
TQtNDA0LjU4LDM0LjYzLTQxLjE0LDEyMC0xNTEuNiwyMDIuNzUtMjQyLjE5LTMuMTMsNy02LjEyLDE
0LjI1LTguOTIsMjEuNjktMjQuMzQsNjQuNDUtMzYuNjcsMTQ0LjMyLTM2LjY3LDIzNy40MSwwLDU2LjU
zLDUuNTgsMTA2LDE2LjU5LDE0Ny4xNEEzMDcuNDksMzA3LjQ5LDAsMCwwLDI4MC45MSw3MjND
MjM3LDgxNi44OCwyMTYuOTMsODkzLjkzLDEzNC41OCw5MDguMDdaIi8+PHBhdGggY2xhc3M9ImN
scy01IiBkPSJNNTgzLjQzLDgxMy43OUM1NjAuMTgsNzI3LjcyLDUxMiw2NjQuMTUsNTEyLDY2NC4xN
XMtNDguMTcsNjMuNTctNzEuNDMsMTQ5LjY0Yy00OC40NS02Ljc0LTEwMC45MS0yNy41Mi0xMzUu
NjYtOTEuMThhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Ny03MS41NGwuMjEtLjMyLjE5LS4zM2M2M
zOC02My42MywxMjYuNC0xOTEuMzcsMTY3LjEyLTI0NS42Niw0MC43MSw1NC4yOCwxMjkuMSwxO
DIsMTY3LjEyLDI0NS42NmwuMTkuMzMuMjEuMzJhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Nyw
3MS41NEM2ODQuMzQsNzg2LjI3LDYzMS44OCw4MDcuMDUsNTgzLjQzLDgxMy43OVoiLz48cGF0a
CBjbGFzcz0iY2xzLTQiIGQ9Ik04ODkuNzUsOTA4YS4zOS4zOSwwLDAsMS0uMjcuMTFoLS4wNkM4M
DcuMDcsODkzLjkzLDc4Nyw4MTYuODgsNzQzLjA5LDcyM2EzMDcuNDksMzA3LjQ5LDAsMCwwLDIwL
jQ1LTU1LjU0YzExLTQxLjExLDE2LjU5LTkwLjYxLDE2LjU5LTE0Ny4xNCwwLTkzLjA4LTEyLjMzLTE3M
y0zNi42Ni0yMzcuNHEtNC4yMi0xMS4xNi04LjkzLTIxLjdjODIuNzUsOTAuNTksMTY4LjEyLDIwMS4wNS
wyMDIuNzUsMjQyLjE5QzEwNDQuNzksNjIwLjU2LDEwMDkuMjcsNzg2Ljg5LDg4OS43NSw5MDhaIi8+
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。



8.4.2.6. introduction 要素

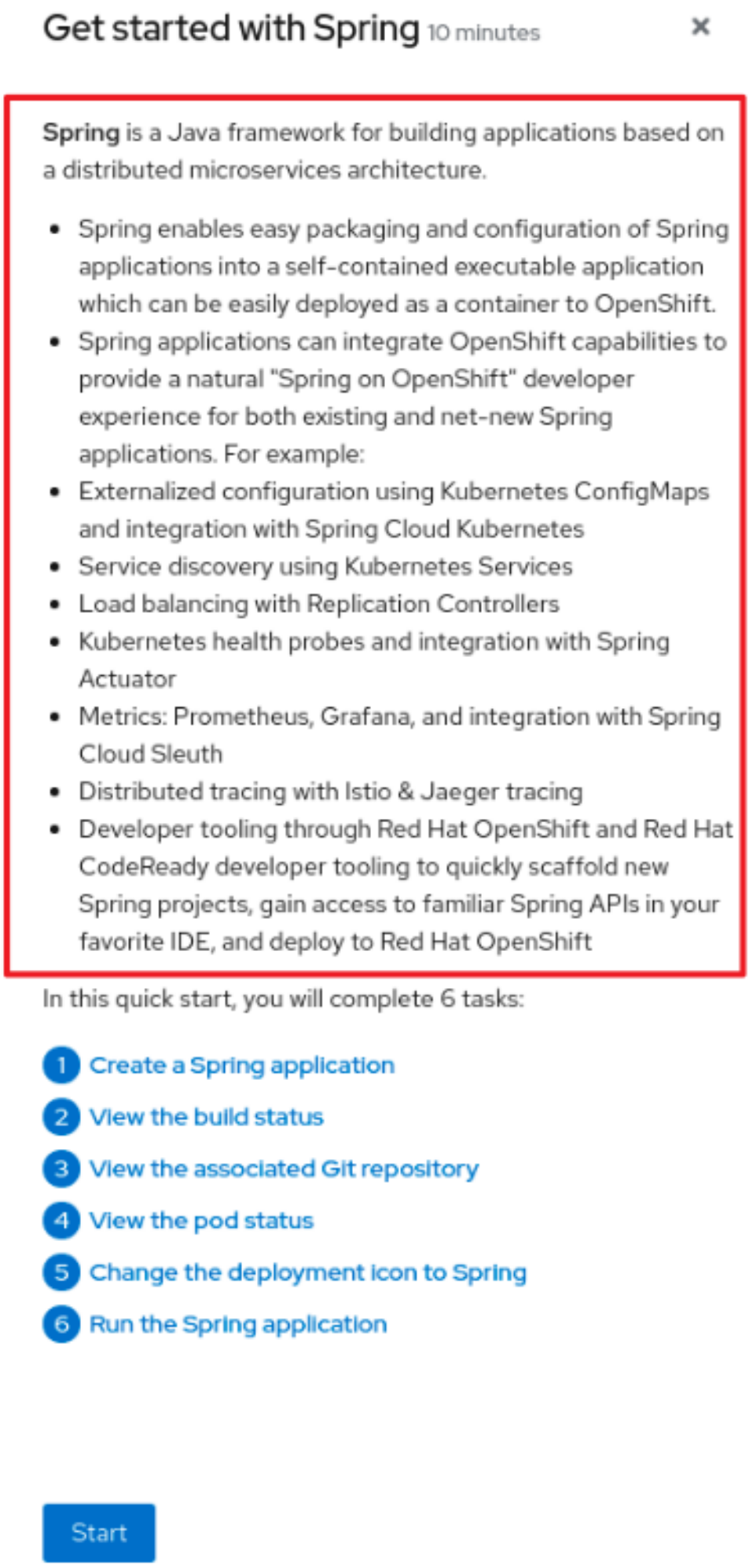
YAML ファイルでの introduction 要素の表示

```
...
  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 & Jaeger tracing

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

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

Web コンソールでの introduction 要素の表示

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


Get started with Spring 10 minutes
Spring is a Java framework for building applications based on a distributed microservices architecture.
Spring enables easy packaging and configuration of Spring applications into a self-contained executable application which can be easily deployed as a container to OpenShift.
Spring applications can integrate OpenShift capabilities to provide a natural "Spring on OpenShift" developer experience for both existing and net-new Spring applications. For example:
Externalized configuration using Kubernetes ConfigMaps and integration with Spring Cloud Kubernetes
Service discovery using Kubernetes Services
Load balancing with Replication Controllers
Kubernetes health probes and integration with Spring Actuator
Metrics: Prometheus, Grafana, and integration with Spring Cloud Sleuth
Distributed tracing with Istio & Jaeger tracing
Developer tooling through Red Hat OpenShift and Red Hat CodeReady developer tooling to quickly scaffold new Spring projects, gain access to familiar Spring APIs in your favorite IDE, and deploy to Red Hat OpenShift
In this quick start, you will complete 6 tasks:
1 Create a Spring application
2 View the build status
3 View the associated Git repository
4 View the pod status
5 Change the deployment icon to Spring
6 Run the Spring application
Start

8.4.3. カスタムアイコンのクイックスタートへの追加

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

手順

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

```
icon: >-


ciIHJvbGU9ImltZyIgdmlld.
```

8.4.4. クイックスタートへのアクセス制限

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

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

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

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

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

8.4.5. 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。以下は例になります。

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

8.4.6. 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。

8.4.7. 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

8.4.7.1. パースペクティブスイッチャー

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

8.4.7.2. 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}}
```

8.4.7.3. 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}}
```

8.4.7.4. 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}}
```

8.4.7.5. 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}}
```

8.4.8. 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 ターミナル Operator をインストールする必要があります。Web ターミナルで実行する Web ターミナルおよびコードスニペットの各種アクションは、Web ターミナル 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 ターミナル 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。

8.4.8.1. 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 ターミナル 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。

8.4.8.2. 複数行コードスニペットの構文

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

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

8.5. クイックスタートのコンテンツガイドライン

8.5.1. 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.

8.5.2. はじめに

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

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

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

  - **正しくない例**:

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

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

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

  - **正しくない例**:

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

8.5.3. タスクの手順

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

タスクのステップを作成する場合は、以下の一般的なガイドラインに従います。

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

    > Click OK.

  - **正しくない例**:

> Click on the OK button.

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

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

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

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

  - **正しくない例**:

    > Hover over the icon in the node.js deployment.

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

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

  - **追加情報**:

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

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

    > In the navigation menu, click Settings.

  - **正しくない例**:

    > In the left-hand menu, click Settings.

- 「灰色の円」など、色でしかアイテムを特定できないようにしないでください。色識別子は、視力制限のあるユーザー、とくに色覚異常のユーザーに役立ちません。その代わりに、ボタンコピーのような名前またはコピーを使用してアイテムを特定します。
  - **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.

8.5.4. 作業モジュールの確認

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

8.5.5. 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。

8.6. 追加リソース

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