OpenShift Container Platform 4.10

Web 控制台

在 OpenShift Container Platform 中使用 web 控制台

Last Updated: 2023-10-18

OpenShift Container Platform 4.10 Web 控制台

在 OpenShift Container Platform 中使用 web 控制台

法律通告

摘要

本文档提供了有关使用和定制 OpenShift Container Platform web 控制台的信息。

目录

第 1 章 WEB 控制台概述

Red Hat OpenShift Container Platform Web 控制台提供了一个图形用户界面，用于视觉化您的项目数据，并执行管理、管理和故障排除任务。Web 控制台在 openshift-console 项目的 control plane 节点上运行。它由一个 **console-operator** pod 管理。支持**管理**员和**开发者**视角。

管理员和**开发者**视角都允许您为 OpenShift Container Platform 创建快速开始指南。快速开始是关于用户任务的指导教程，可用于熟悉应用程序、Operator 或其他产品。

1.1. 关于 WEB 控制台中的 ADMINISTRATOR 视角

Administrator 视角可让您查看集群清单、容量、常规和特定使用信息以及重要事件的流，它们可帮助您简化计划和故障排除任务。项目管理员和开发人员可以使用**管理**员视角。

集群管理员还可在 OpenShift Container Platform 4.7 及之后的版本中为 web 终端 Operator 打开内嵌的命令行终端实例。

注意

显示的默认 Web 控制台视角取决于用户的角色。如果用户被视为管理员，则默认显示**管理**员视角。

管理员视角提供特定于管理员用例的工作流，例如：

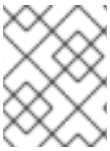
- 管理工作负载、存储、网络和集群设置。
- 使用 Operator Hub 安装和管理 Operator。
- 添加允许用户通过角色和角色绑定登录和管理用户访问权限的身份提供程序。
- 查看和管理各种高级设置，如集群更新、部分集群更新、集群 Operator、自定义资源定义 (CRD)、角色绑定和资源配额。
- 访问和管理监控功能，如指标、警报和监控仪表板。
- 查看并管理有关集群的日志记录、指标和高状态信息。
- 在 OpenShift Container Platform 中，以视觉化的方式和与**管理**员视角关联的应用程序、组件和服务交互。

1.2. 关于 WEB 控制台中的开发者视角

开发者视角提供了几个用来部署应用程序、服务和数据库的内置方法。在**开发者**视角中，您可以：

- 查看组件上滚动和重新创建推出部署的实时视觉化。
- 查看应用状态、资源利用率、项目事件流和配额消耗。
- 将您的项目与他人共享。
- 通过在项目上运行 Prometheus Query Language(PromQL)查询并查看图表中呈现的指标来排除应用程序的问题。此指标数据提供有关集群以及要监控的任何用户定义工作负载的状态信息。

集群管理员也可以在 OpenShift Container Platform 4.7 及之后的版本中的 Web 控制台中打开内嵌的命令行终端实例。



注意

显示的默认 Web 控制台视角取决于用户的角色。如果用户是开发人员，则 **Developer** 视角会被默认显示。

Developer 视角提供开发人员用例特有的工作流，比如：

- 通过导入现有代码基、镜像和容器文件在 OpenShift Container Platform 中创建和部署应用程序。
- 在一个项目中，以可视的形式和与其关联的应用程序、组件和服务进行交互，并监控它们的部署和构建状态。
- 在应用程序中对组件进行分组，并在应用程序内部及跨应用程序间连接组件。
- 集成无服务器功能（技术预览）。
- 使用 Eclipse Che 创建开发平台来编辑应用程序代码。

您可以使用 **Topology** 视图来显示项目的应用程序、组件和工作负载。如果项目中没有工作负载，则 **Topology** 视图将显示一些创建或导入它们的链接。您还可以使用 **Quick Search** 来直接导入组件。

其它资源

如需有关在 **Developer** 视角中使用 Topology 视图的更多信息，请参阅**使用 Topology 视图查看应用程序组成**。

1.3. 访问视角

您可以通过 web 控制台访问 **Administrator** 和 **Developer** 视角，如下所示：

先决条件

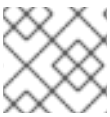
要访问视角，请确保您已登陆到 web 控制台。您的默认视角由用户权限自动决定。对有权访问所有项目的用户选择 **Administrator** 视角，而对于对自己项目具有有限访问权限的用户选择 **Developer** 视角

其它资源

有关更改视角的更多信息，请参阅添加用户首选项。

流程

1. 使用视角切换器切换到 **Administrator** 或 **Developer** 视角。
2. 从 **Project** 下拉列表中选择一个现有项目。您也可以从此下拉菜单创建新项目。



注意

您只能以 **cluster-admin** 用户身份使用视角切换器。

其他资源

- 了解有关集群管理员的更多信息
- **管理员**视角概述

- 使用 **Developer** 视角在 OpenShift Container Platform 中创建并部署应用程序
- 使用 **Topology** 视图查看项目中的应用程序，验证应用程序的部署状态，并与应用程序进行交互。
- 查看集群信息
- 配置Web控制台
- 自定义 Web 控制台
- 使用 web 终端
- 创建快速启动指南
- 禁用Web控制台

第 2 章 访问WEB控制台

OpenShift Container Platform Web控制台是可从Web浏览器访问的用户界面。开发人员可以使用Web控制台来直观地浏览并管理项目的内容。

2.1. 先决条件

- 必须启用JavaScript才能使用Web控制台。为获得最佳体验，请使用支持WebSockets的Web浏览器。
- 在为集群创建支持基础结构之前，请参阅OpenShift Container Platform 4.x Tested Integrations页。

2.2. 了解和访问WEB控制台

Web控制台作为一个pod 在主服务器（master）上运行。这个 pod 提供了运行Web控制台所需的静态环境。当使用 **openshift-install create cluster** 成功安装 OpenShift Container Platform 后，在安装程序的 CLI 输出中找到已安装集群的 Web 控制台的 URL 及登录凭证。例如：

输出示例

```
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>
```

使用这些信息登录并访问Web控制台。

对于不是您安装的、已存在的集群，可以使用 **oc whoami --show-console** 查看 web 控制台 URL。

其他资源

- 使用 Web 控制台启用功能集

第 3 章 使用 OPENSHIFT CONTAINER PLATFORM DASHBOARD 获取集群信息

OpenShift Container Platform 仪表板（dashboard） 包括了集群的高级别信息。在 OpenShift Container Platform web 控制台中通过 **Home → Dashboards → Overview** 访问它。

OpenShift Container Platform 仪表板提供各种集群信息，被分别显示在独立的仪表板卡中。

3.1. 关于 OPENSHIFT CONTAINER PLATFORM 仪表板页

OpenShift Container Platform 仪表板由以下各卡组成：

- **Details** 提供有关信息型集群详情的简单概述。
  状态包括 **ok**、**error**、**warning**、**in progress** 和 **unknown**。资源可添加自定义状态名称。
  - 集群 ID
  - 提供者
  - 版本
- **Cluster Inventory** 详细列出资源数目和相关状态。这在通过干预解决问题时非常有用，其中包含以下相关信息：
  - 节点数
  - pod 数量
  - 持久性存储卷声明
  - 集群中的裸机主机，根据其状态列出（只在 **metal3** 环境中可用）。
- **Cluster Capacity** 图表可帮助管理员了解在什么时候集群需要额外的资源。此表包含一个内环和一个外环。内环显示当前的消耗，外环显示为资源配置的阈值，其中包括以下信息：
  - CPU 时间
  - 内存分配
  - 所消耗的存储
  - 所消耗的网络资源
- **Cluster Utilization** 显示在指定时间段内各种资源的能力，以帮助管理员了解高资源消耗的范围和频率。
- **Events** 列出了与集群中最近活动相关的消息，如创建 pod 或虚拟机迁移到另一台主机。
- **Top Consumers** 可帮助管理员了解集群资源是如何被消耗的。点一个资源可以进入一个包括详细信息的页面，它列出了对指定集群资源（CPU 、内存或者存储）消耗最多的 Pod 和节点。

第 4 章 添加用户首选项

您可以更改您的配置集的默认首选项以满足您的要求。您可以设置默认项目、拓扑视图（图形/列表）、编辑介质（格式或 YAML）以及语言首选项。

对用户首选项所做的更改会自动保存。

4.1. 设置用户首选项

您可以为集群设置默认用户首选项。

流程

1. 使用您的登录凭证登录到 OpenShift Container Platform web 控制台。
2. 使用 masthead 访问用户配置文件下的用户首选项。
3. 在 **General** 部分中：
   a. 在 **Perspective** 字段中，您可以设置要登录到的默认视角。您可以根据需要选择 **Administrator** 或 **Developer** 视角。如果未选择透视图，您会登录到您最近看到的视角。
   b. 在 **Project** 字段中，选择一个您要处理的项目。每次登录时，控制台都将默认为项目。
   c. 在 **Topology** 字段中，您可以将拓扑视图设置为 default 到 graph 或 list 视图。如果未选中，控制台将默认为您使用的最后一个视图。
   d. 在 **Create/Edit resource method** 字段中，您可以设置创建或编辑资源的首选。如果表单和 YAML 选项都可用，控制台默认为您的选择。
4. 在 Language 部分中，选择 **Default 浏览器语言**以使用默认的浏览器语言设置。否则，请选择您要用于控制台的语言。

第 5 章 在OPENSHIFT CONTAINER PLATFORM中配置WEB控制台

您可以修改OpenShift Container Platform Web控制台以设置注销重定向URL，或禁用控制台。

5.1. 先决条件

- 部署一个OpenShift Container Platform集群。

5.2. 配置WEB控制台

您可以通过编辑**console.config.openshift.io**资源来配置Web控制台设置。

- 编辑**console.config.openshift.io**资源：

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

以下是控制台的资源定义示例：

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

(1) 指定用户注销后，Web控制台要加载页面的URL。如果未指定，则用户将会返回到Web控制台的登录页面。通过指定**logoutRedirect** URL，用户可以使用身份供应商的单点注销（SLO）功能销毁其单点登录会话。

(2) Web控制台URL。要将它更新为自定义值，请参阅**自定义 Web 控制台 URL**。

5.3. 在 WEB 控制台中禁用快速启动

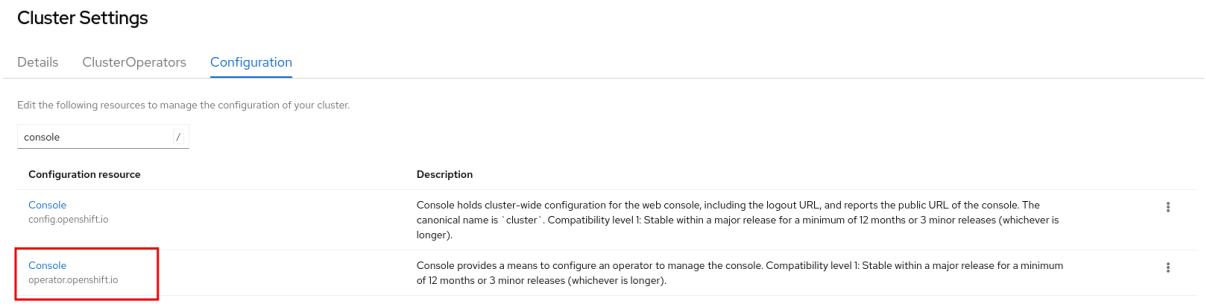
您可以使用 Web 控制台的 **Administrator** 视角来禁用一个或多个快速启动。

先决条件

- 有集群管理员权限并登录到 web 控制台。

流程

1. 在 **Administrator** 视角中，进入 **Administration → Cluster Settings**。
2. 在 **Cluster Settings** 页面中，点 **Configuration** 选项卡。
3. 在 **Configuration** 页面中，点带有描述 **operator.openshift.io** 的 **Console** 配置资源。



4. 从 **Action** 下拉列表中，选择 **Customize**，以打开 **Cluster 配置**页面。

5. 在 **General** 选项卡中，在 **Quick start** 部分中，您可以在 **Enabled** 或 **Disabled** 列表中选择项目，并使用箭头按钮将它们从一个列表中移到另一个列表中。

   - 要启用或禁用单个快速启动，请点快速启动，然后使用单个箭头按钮将快速启动移到适当的列表中。
   - 要一次启用或禁用多个快速启动，请按 Ctrl 并点击您要移动的快速启动。然后，使用单箭头按钮将快速启动移到适当的列表中。
   - 要一次启用或禁用所有快速启动，请点双箭头按钮将所有快速开始移到适当的列表中。

第 6 章 在 OPENSHIFT CONTAINER PLATFORM 中定制 WEB 控制台

您可以对 OpenShift Container Platform web 控制台进行定制，如设置自定义徽标、产品名、链接、通知标语和命令行下载。这在您需要定制 Web 控制台以满足具体公司或政府要求时特别有用。

6.1. 添加自定义徽标和产品名称

您可以通过添加自定义徽标或自定义产品名称来创建自定义品牌。因为这些设置相互独立，因此可以两者都设置或只设置其中的一个。

先决条件

- 您必须具有管理员特权。
- 创建一个要使用的徽标文件。徽标可以是通用图像格式的文件，包括 GIF 、JPG 、PNG 或 SVG ，并有 **max-height** 为 **60px**的限制。

流程

1. 在 **openshift-config** 命名空间中将您的徽标文件导入到配置映射中：

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

提示

您还可以应用以下 YAML 来创建配置映射：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: console-custom-logo
  namespace: openshift-config
binaryData:
  console-custom-logo.png: <base64-encoded_logo> ... 1
```

1 提供有效的 base64 编码徽标。

2. 编辑 web 控制台的 Operator 配置使其包含 **customLogoFile** 和 **customProductName**：

```
$ 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
```

更新 Operator 配置后，它将会把自定义的 logo 配置映射同步到控制台命名空间中，并将其挂载到 console pod 并重新部署。

3. 检查操作是否成功。如果有任何问题，控制台集群 Operator 将报告 **Degraded** 状态，控制台 Operator 配置也会报告 **CustomLogoDegraded** 状态，但状态类似于 **KeyOrFilenameInvalid** 或 **NoImageProvided**。
运行以下命令检查 **clusteroperator**：

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

运行以下命令检查 console Operator 配置：

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

6.2. 在 WEB 控制台中创建自定义链接

先决条件

- 您必须具有管理员特权。

流程

1. 在 **Administration → Custom Resource Definitions** 中点 **ConsoleLink**。
2. 选择 **Instances** 标签
3. 点击 **Create Console Link** 并编辑文件：

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

1 有效的位置设置为 **HelpMenu**、**UserMenu**、**ApplicationMenu** 和 **NamespaceDashboard**。

要使自定义链接出现在所有命名空间中，请按照以下示例操作：

```
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
```

要使自定义链接只出现在某些命名空间中，请按照以下示例操作：

```
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
```

要使自定义链接出现在应用程序菜单中，请按照以下示例操作：

```
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. 点击 **Save** 以应用您的更改。

6.3. 自定义控制台路由

对于 **console** 和 **downloads** 路由，自定义路由功能使用 **ingress** 配置路由 API。如果 **console** 自定义路由在 **ingress** 配置和 **console-operator** 配置中都设置时，新的 **ingress** 配置自定义路由配置有高的优先级。带有 **console-operator** 配置的路由配置已弃用。

6.3.1. 自定义控制台路由

您可以通过在集群 **Ingress** 配置的 **spec.componentRoutes** 字段中设置自定义主机名和 TLS 证书来自定义控制台路由。

先决条件

- 已使用具有管理特权的用户身份登录集群。

- 您已在 **openshift-config** 命名空间中创建了包含 TLS 证书和密钥的 secret。如果自定义主机名后缀的域与集群域后缀不匹配，则需要此项。如果后缀匹配，secret 是可选的。

  提示

  您可以使用 **oc create secret tls** 命令创建 TLS secret。

流程

1. 编辑集群 **Ingress** 配置：

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

2. 设置自定义主机名以及可选的服务证书和密钥：

```
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. 自定义主机名.
2. 对 **openshift-config** 命名空间中的 secret 的引用，该 secret 包含 TLS 证书 (**tls.crt**) 和密钥 (**tls.key**)。如果自定义主机名后缀的域与集群域后缀不匹配，则需要此项。如果后缀匹配，secret 是可选的。

3. 保存文件以使改变生效。

6.3.2. 自定义下载路由

您可以通过在集群 **Ingress** 配置的 **spec.componentRoutes** 字段中设置自定义主机名和 TLS 证书来自定义下载路由。

先决条件

- 已使用具有管理特权的用户身份登录集群。
- 您已在 **openshift-config** 命名空间中创建了包含 TLS 证书和密钥的 secret。如果自定义主机名后缀的域与集群域后缀不匹配，则需要此项。如果后缀匹配，secret 是可选的。

  提示

  您可以使用 **oc create secret tls** 命令创建 TLS secret。

流程

1. 编辑集群 **Ingress** 配置：

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

2. 设置自定义主机名以及可选的服务证书和密钥：

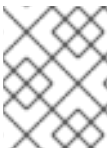
```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  componentRoutes:
    - name: downloads
      namespace: openshift-console
      hostname: <custom_hostname> 1
      servingCertKeyPairSecret:
        name: <secret_name> 2
```

1. 自定义主机名.
2. 对 **openshift-config** 命名空间中的 secret 的引用，该 secret 包含 TLS 证书 (**tls.crt**) 和密钥 (**tls.key**)。如果自定义主机名后缀的域与集群域后缀不匹配，则需要此项。如果后缀匹配，secret 是可选的。

3. 保存文件以使改变生效。

6.4. 自定义登录页面

使用自定义登录页面创建服务条款信息。如果您使用第三方登录提供程序（如 GitHub 或 Google），在将用户信任并期望它重定向到认证提供程序之前，自定义登录页面也会很有用。您还可以在验证过程中显示自定义的错误页。



注意

自定义错误模板仅限于使用重定向的身份提供程序（IDP），如请求标头和基于 OIDC 的操作。它对使用直接密码身份验证的 IDP （如 LDAP 和 htpasswd）没有影响。

先决条件

- 您必须具有管理员特权。

流程

1. 运行以下命令来创建您可以修改的模板：

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

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

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

2. 创建 secret:

```
$ 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. 运行：

```
$ oc edit oauths cluster
```

4. 更新规格：

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

运行 **oc explain oauths.spec.templates** 以了解选项。

6.5. 为外部日志链接定义模板

如果您连接到可帮助您浏览日志的服务，但需要以特定的方式生成 URL，则可以为链接定义一个模板。

先决条件

- 您必须具有管理员特权。

流程

1. 在 **Administration → Custom Resource Definitions** 中点 **ConsoleExternalLogLink**。
2. 选择 **Instances** 标签
3. 点击 **Create Console External Log Link** 并编辑文件：

```
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. 创建自定义通知标语

先决条件

- 您必须具有管理员特权。

流程

1. 在 **Administration → Custom Resource Definitions**中点 **ConsoleNotification**。
2. 选择 **Instances** 标签
3. 点击 **Create Console Notification** 并编辑文件：

```
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) 有效的位置设置为 **BannerTop**、**BannerBottom** 和 **BannerTopBottom**。

4. 点 **Create** 以应用您的更改。

6.7. 自定义 CLI 下载

您可以使用自定义链接文本和 URL 来配置用于下载 CLI 的链接。它们可以直接指向软件包的文件或提供软件包的外部页面。

先决条件

- 您必须具有管理员特权。

流程

1. 进入 **Administration → Custom Resource Definitions**。
2. 从 Custom Resource Definitions (CRDs) 列表中选 **ConsoleCLIDownload** 。
3. 点 **YAML** 标签页，然后进行编辑：

```
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. 点 **Save** 按钮。

6.8. 在 KUBERNETES 资源中添加 YAML 示例

您可以随时动态地将 YAML 示例添加到任何 Kubernetes 资源中。

先决条件

- 您必须具有集群管理员特权。

流程

1. 在 **Administration → Custom Resource Definitions** 中点 **ConsoleYAMLSample**。
2. 点 **YAML** 并编辑该文件：

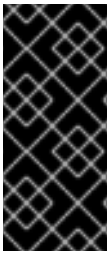
```
apiVersion: console.openshift.io/v1
kind: ConsoleYAMLSample
metadata:
  name: example
spec:
  targetResource:
    apiVersion: batch/v1
    kind: Job
  title: Example Job
  description: An example Job YAML sample
  yaml: |
    apiVersion: batch/v1
    kind: Job
    metadata:
      name: countdown
    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
```

使用 **spec.snippet** 表示 YAML 样本不是完整的 YAML 资源定义，而是可在用户光标处的现有 YAML 文档中插入的片段。

3. 点击 **Save**。

第 7 章 在 OPENSHIFT CONTAINER PLATFORM WEB 控制台中添加动态插件

您可以在运行时载入的集群中创建和部署动态插件。



重要

创建动态插件只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

7.1. 关于动态插件

动态插件允许您在运行时为接口添加自定义页面和其他扩展。**ConsolePlugin** 自定义资源使用控制台注册插件，集群管理员在 **console-operator** 配置中启用了插件。

7.1.1. 主要特性

通过动态插件，您可以对 OpenShift Container Platform 体验进行以下自定义：

- 添加自定义页面。
- 增加了除管理员和开发人员之外的其他视角。
- 添加导航项。
- 在资源页面中添加制表符和操作。

7.1.2. 常规指南

在创建插件时，请遵循以下常规准则：

- 构建和运行插件需要 **Node.js** 和 **yarn**。
- 为您的 CSS 类名称加上插件名称前缀，以避免冲突。例如，**my-plugin__heading** 和 **my-plugin__icon**。
- 与其他控制台页面保持一致的外观、感觉和行为。
- 在创建插件时遵循 react-i18next 指南。您可以使用类似以下示例中的 **useTranslation** hook:

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

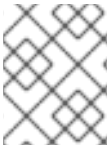
- 避免可能影响插件组件之外的标记的选择器，如元素选择器。这些不是 API，可能随时更改。使用它们可能会破坏插件。

PatternFly 4 指南
在创建插件时，请按照以下使用 PatternFly 的准则进行以下操作：

- 使用 PatternFly4 组件和 PatternFly CSS 变量。SDK 提供了核心 PatternFly 组件。使用 PatternFly 组件和变量可帮助您的插件在将来的控制台版本中保持一致。
- 您可以按照 PatternFly 的可访问性基础，使您的插件能被访问。
- 避免使用其他 CSS 库，如 Bootstrap 或 Tailwind。它们可能会与 PatternFly 冲突，不会与控制台的外观和感觉相匹配。

7.2. 动态插件入门

您可以对 OpenShift Container Platform Web 控制台进行不同的自定义配置。设置您的环境以编写新的 OpenShift 控制台动态插件，并在 **Pod 详情**页面中添加标签页作为插件的一个示例扩展。



注意

OpenShift Container Platform Web 控制台在一个连接到您登录的集群的容器中运行。有关在创建自己的前测试插件的信息，请参阅"运行您的动态插件"。

先决条件

- 确保安装了 **Node.js**。
- 确定安装了 **yarn**。

流程

1. 在新标签页中，打开 **console-plugin-template** 存储库，其中包含用于在新标签页中创建插件的模板。

重要

红帽不支持自定义插件代码。对于插件，只有合作社区支持。

2. 点 **Use this template → Create new repository** 从模板创建 GitHub 存储库。
3. 使用插件的名称替换新存储库。
4. 将新存储库克隆到本地机器，以便您可以编辑代码。
5. 编辑 **package.json** 文件，将插件的元数据添加到 **consolePlugin** 声明中。例如：

```
"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. 更新插件的名称。
2. 更新版本。
3. 更新插件的显示名称。
4. 使用有关插件的同步更新描述。

6. 在 **console-extensions.json** 文件中添加以下内容：

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

7. 编辑 **package.json** 文件以包括以下更改：

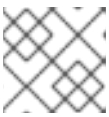
```
    "exposedModules": {
      "ExamplePage": "./components/ExamplePage",
      "ExampleTab": "./components/ExampleTab"
    }
```

8. 通过创建新文件 **src/components/ExampleTab.tsx** 并添加以下脚本，在 **Pod** 页面上的一个新自定义标签页中写入信息：

```
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. 使用插件名称作为 Helm 发行版本名称安装 Helm Chart，或由 **-n** 命令行选项指定的现有命名空间，以便在集群中部署插件。使用以下命令，提供 **plugin.image** 参数中镜像的位置：

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



注意

有关在集群中部署插件的更多信息，请参阅"在集群中部署插件"。

验证

- 访问 **Pod** 页面查看添加的选项卡。

7.3. 使用 DOCKER 构建镜像

要在集群中部署插件，您需要构建镜像并将其推送到镜像 registry。

流程

1. 使用以下命令构建镜像：

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

2. 可选：如果要测试您的镜像，请运行以下命令：

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

3. 运行以下命令推送镜像：

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

7.4. 运行您的动态插件

您可以使用本地开发环境运行插件。OpenShift 控制台在连接到您登录的集群的容器中运行。

先决条件

- 已安装 OpenShift CLI (**oc**)。
- 您必须有一个 OpenShift 集群正在运行。
- 必须至少安装了 Podman 的 Docker 或 v3.2.0。

流程

- 在克隆存储库的本地目录中打开两个终端窗口。
  a. 在第一个终端中运行以下命令：

```
$ yarn install
```

```
$ yarn run start
```

  b. 在第二个终端窗口中运行以下命令：

```
$ oc login
```

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

验证

- 访问 本地主机 以查看正在运行的插件。

7.5. 在集群中部署插件

在将镜像推送到 registry 后，您可以将插件部署到集群中。

流程

1. 要将插件部署到集群中，请使用插件名称作为 Helm 发行版本名称安装 Helm chart，作为 Helm 发行版本名称，或由 **-n** 命令行选项指定的现有命名空间。使用以下命令，提供 **plugin.image** 参数中镜像的位置：

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

其中：

n <my-plugin-namespace>

指定要将插件部署到的现有命名空间。

--create-namespace

可选：如果部署到新命名空间，请使用此参数。

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

指定 **plugin.image** 参数中镜像的位置。

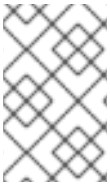
2. 可选：您可以使用 **charts/openshift-console-plugin/values.yaml** 文件中的一组支持的参数来指定任何其他参数。

```
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
```

验证

- 您可以通过 **Administration → Cluster Settings → Configuration → Console operator.openshift.io → Console plugins** 或 **Overview** 页中查看已启用的插件列表。



注意

显示新插件配置可能需要几分钟时间。如果没有看到插件，则在最近启用了插件时，可能需要刷新浏览器。如果您在运行时收到任何错误，请在浏览器开发人员工具中检查 JS 控制台，以查看插件代码中的任何错误。

7.6. 其他资源

- 了解 Helm

第 8 章 WEB 终端

8.1. 安装 WEB 终端

您可以使用 OpenShift Container Platform OperatorHub 中列出的 Web Terminal Operator 来安装 Web 终端。安装 Web Terminal Operator 时，会自动安装命令行配置（如 **DevWorkspace** CRD）所需的自定义资源定义（CRD）。打开 web 终端时，web 控制台会创建所需的资源。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- 有集群管理员权限。

流程

1. 在 Web 控制台的 **Administrator** 视角中，导航到 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 复选框在目录中搜索 Web Terminal Operator，然后点 **Web Terminal** 标题。
3. 参阅 **Web Terminal** 页面中有关 Operator 的简单描述，然后点击 **Install**。
4. 在 **Install Operator** 页面中，保留所有字段的默认值。
   - **Update Channel** 菜单中的 **fast** 选项启用 Web Terminal Operator 最新版本的安装。
   - **Installation Mode** 菜单中的 **All namespaces on the cluster** 选项可让 Operator 监视并可供集群中的所有命名空间使用。
   - **Installed Namespace** 菜单中的 **openshift-operators** 选项会在默认的 **openshift-operators** 命名空间中安装 Operator。
   - **Approval Strategy** 菜单中的 **Automatic** 选项确保以后对 Operator 的升级由 Operator Lifecycle Manager 自动处理。
5. 点 **Install**。
6. 在 **Installed Operators** 页面中，点 **View Operator** 来验证 **Installed Operators** 页面中是否列出了 Operator。

   注意

   Web Terminal Operator 将 DevWorkspace Operator 安装为依赖项。

7. 安装 Operator 后，刷新页面以查看控制台 masthead 中的命令行终端图标()。

8.2. 使用 WEB 终端

您可以在 web 控制台中启动内嵌的命令行终端实例。此终端实例预安装了与集群交互的通用 CLI 工具，如 **oc**、**kubectl**、**odo**、**kn**、**tkn**、**helm**、**kubens**、**subctl** 和 **kubectx**。它还包含正在处理的项目的上下文，并自动记录您使用凭证的项目。

8.2.1. 访问 Web 终端

安装 Web Terminal Operator 后，您可以访问 Web 终端。您可以从在终端中运行的命令列表中选择这些命令，以重新运行这些命令。这些命令可在多个终端会话中保留。Web 终端保持打开，直到您关闭浏览器窗口或标签页。

先决条件

- 您可以访问 OpenShift Container Platform 集群，并登录到 web 控制台。
- 在集群中安装了 Web Terminal Operator。

流程

1. 要启动 web 终端，请在控制台的 masthead 中点命令行终端图标()。在 **Command line terminal** 窗格中会显示 web 终端实例。此实例使用您的凭证自动登录。
2. 从 Project 下拉列表中选择创建 **DevWorkspace** CR 的 项目。默认情况下会选择当前项目。

注意

- 只有在不存在 **DevWorkspace** CR 时才会创建 DevWorkspace CR。
- **openshift-terminal** 项目是集群管理员使用的默认项目。它们没有选择其他项目的选项。

3. 点 **Start** 使用所选项目初始化 Web 终端。初始化 web 终端后，您可以在 web 终端中使用预安装的 CLI 工具，如 **oc**、**kubectl**、**odo**、**kn**、**tkn** 、 **helm**、kubens、**subctl** 和 **kubectx**。

8.3. 对 WEB 终端进行故障排除

8.3.1. Web 终端和网络策略

如果集群配置了网络策略，web 终端可能无法启动。要初始化 Web 终端实例，Web Terminal Operator 必须与 Web 终端的 pod 通信以验证它是否正在运行，OpenShift Container Platform Web 控制台需要发送信息才能在终端中自动登录到集群。如果任一步骤失败，Web 终端将无法初始化，终端面板看似处于加载状态。

要避免这个问题，请确保用于终端的网络策略允许从 **openshift-console** 和 **openshift-operators** 命名空间进行入站数据。

8.4. 卸载 WEB 终端

卸载 Web Terminal Operator 不会删除安装 Operator 时创建的任何自定义资源定义 (CRD) 或受管资源。为了安全起见，您必须手动卸载这些组件。通过删除这些组件，您可以保存集群资源，因为在卸载 Operator 时终端不会闲置。

卸载 web 终端需要两步：

1. 卸载 Web Terminal Operator 和安装 Operator 时添加的相关自定义资源 (CR)。
2. 卸载 DevWorkspace Operator 及其作为 Web Terminal Operator 依赖项添加的相关自定义资源。

8.4.1. 删除 Web Terminal Operator

您可以通过删除 Web Terminal Operator 和 Operator 使用的自定义资源来卸载 web 终端。

先决条件

- 您可以使用集群管理员权限访问 OpenShift Container Platform 集群。
- 已安装 **oc** CLI。

流程

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Operators → Installed Operators**。
2. 滚动过滤器列表或在 **Filter by name** 框中输入关键字以查找 Web Terminal Operator。
3. 点击 Web Terminal Operator 的 Options 菜单  ，然后选择 **Uninstall Operator**。
4. 在 **Uninstall Operator** 确认对话框中，点 **Uninstall** 从集群中删除 Operator、Operator 部署和 pod。Operator 会停止运行，并且不再接收更新。
5. 删除自定义资源：

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

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

8.4.2. 删除 DevWorkspace Operator

要完全卸载 web 终端，还必须删除 DevWorkspace Operator 和 Operator 使用的自定义资源。

重要

DevWorkspace Operator 是一个独立 Operator，可能需要作为集群中安装的其他 Operator 的依赖项。只有在确保不再需要 DevWorkspace Operator 时，才按照以下步骤操作。

先决条件

- 您可以使用集群管理员权限访问 OpenShift Container Platform 集群。
- 已安装 **oc** CLI。

流程

1. 删除 Operator 使用的 **DevWorkspace** 自定义资源，以及任何相关的 Kubernetes 对象：

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

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

-

> **警告**
>
> 如果此步骤未完成，则终结器很难完全卸载 Operator。

2. 删除 Operator 使用的 CRD：

> **警告**
>
> DevWorkspace Operator 提供了使用转换 Webhook 的自定义资源定义 (CRD)。无法删除这些 CRD 可能会导致集群中的问题。

```
$ 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. 验证所有涉及的自定义资源定义都已移除。以下命令不应该显示任何输出：

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

4. 删除 **devworkspace-webhook-server** 部署、变异并验证 Webhook：

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

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

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

> **注意**
>
> 如果您在没有删除变异并验证 Webhook 的情况下删除 **devworkspace-webhook-server** 部署，则无法使用 **oc exec** 命令在集群中的容器中运行命令。删除 Webhook 后，您可以再次使用 **oc exec** 命令。

5. 删除任何剩余的服务、secret 和配置映射。取决于具体的安装，以下命令中包含的一些资源可能不存在。

```
$ 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. 卸载 DevWorkspace Operator：

   a. 在 web 控制台的 **Administrator** 视角中，导航到 **Operators → Installed Operators**。

   b. 滚动过滤器列表或在 **Filter by name** 框中输入关键字以查找 DevWorkspace Operator。

   c. 点 Operator 的 Options 菜单  ，然后选择 **Uninstall Operator**。

   d. 在 **Uninstall Operator** 确认对话框中，点 **Uninstall** 从集群中删除 Operator、Operator 部署和 pod。Operator 会停止运行，并且不再接收更新。

第 9 章 在OPENSHIFT CONTAINER PLATFORM中禁用WEB控制台

您可以禁用OpenShift Container Platform Web控制台。

9.1. 先决条件

- 部署一个OpenShift Container Platform集群。

9.2. 禁用WEB控制台

您可以通过编辑**consoles.operator.openshift.io**资源来禁用Web控制台。

- 编辑 **consoles.operator.openshift.io** 资源：

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

以下示例显示了资源中可以修改的参数：

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
spec:
  managementState: Removed (1)
```

(1) 将**managementState**参数值设置为**Removed**以禁用Web控制台。此参数的其他有效值是**Managed**（启用由集群控制的控制台），**Unmanaged**（启用由用户控制管理的Web控制台）。

第 10 章 在 WEB 控制台中创建快速启动指南

如果您要为 OpenShift Container Platform Web 控制台创建快速启动指南，请按照以下步骤保留所有快速启动的用户体验。

10.1. 了解快速开始

快速开始是用户任务的指导教程。在 Web 控制台中，您可以在 **Help** 菜单下快速启动访问。它们在使用应用程序、Operator 或其他产品时特别有用。

快速开始主要由任务和步骤组成。每个任务都有多个步骤，每个快速开始都有多个任务。例如：

- 任务 1
  - 第 1 步
  - 第 2 步
  - 第 3 步
- 任务 2
  - 第 1 步
  - 第 2 步
  - 第 3 步
- 任务 3
  - 第 1 步
  - 第 2 步
  - 第 3 步

10.2. 快速启动用户工作流

当您与现有快速启动指南交互时，这是预期的工作流体验：

1. 在 **Administrator** 或 **Developer** 视角中，点击 **Help** 图标并选择 **Quick Starts**。
2. 点快速启动卡。
3. 在出现的面板中点 **Start**。
4. 完成屏幕的说明，然后点 **Next**。
5. 在出现 **Check your work** 模块时，回答问题以确认您成功完成了该任务。
   a. 如果您选择 **Yes**，点 **Next** 来继续到下一个任务。
   b. 如果您选择 **No**，重复任务说明并再次检查您的工作。
6. 重复以上第 1 到 6 步，以便快速完成剩余的任务。
7. 完成最后的任务后，点 **Close** 关闭快速启动。

10.3. 快速启动组件

快速开始由以下部分组成：

- **Card**：提供快速启动基本信息的 catalog 标题，其中包括标题、描述、时间提交和完成状态
- **Introduction**：概述快速开始的目标和任务
- **Task headings**：快速启动中每个任务的超链接标题
- **Check your work module**：一个模块用户使用一个模块来确认他们成功完成了任务，然后到快速启动的下一个任务为止
- **Hints**：帮助用户识别产品特定部分的动画
- **Buttons**
  - **Next and back buttons**：用来浏览快速开始任务里的步骤和模块的按钮
  - **Final screen buttons**：关闭快速启动，返回到快速启动中的先前任务，并查看所有快速启动

快速启动的主要内容包括以下部分：

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

10.4. 快速开始

OpenShift Container Platform 引入了由 **ConsoleQuickStart** 对象定义的快速启动自定义资源。operator 和管理员可以使用此资源来快速使用集群。

先决条件

- 您必须具有集群管理员特权。

流程

1. 要创建新快速启动，请运行：

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

2. 运行：

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

3. 根据本文档中介绍的指南更新 YAML 文件。
4. 保存您的编辑。

10.4.1. 查看快速启动 API 文档

流程

- 要查看快速启动 API 文档，请运行：

```
$ oc explain consolequickstarts
```

运行 **oc explain -h** 以了解有关 **oc explain** 使用的更多信息。

10.4.2. 将快速启动 CR 中的元素映射到快速启动 CR

本节帮助您视觉地映射快速启动自定义资源（CR）的部分，使其出现在 web 控制台中快速启动的自定义资源（CR）中。

10.4.2.1. conclusion 元素

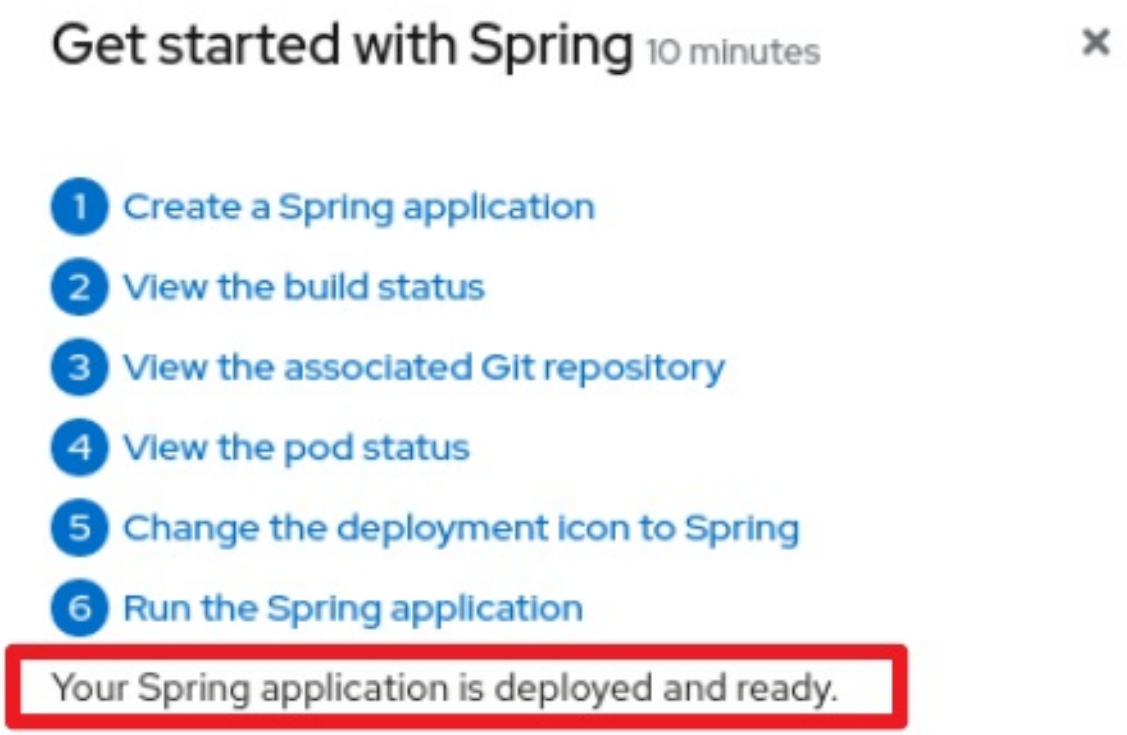
查看 YAML 文件中的 conclusion 元素

```
...
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 文本

在 web 控制台中查看 conclusion 元素

最后会出现在快速开始的最后部分。



10.4.2.2. description 元素

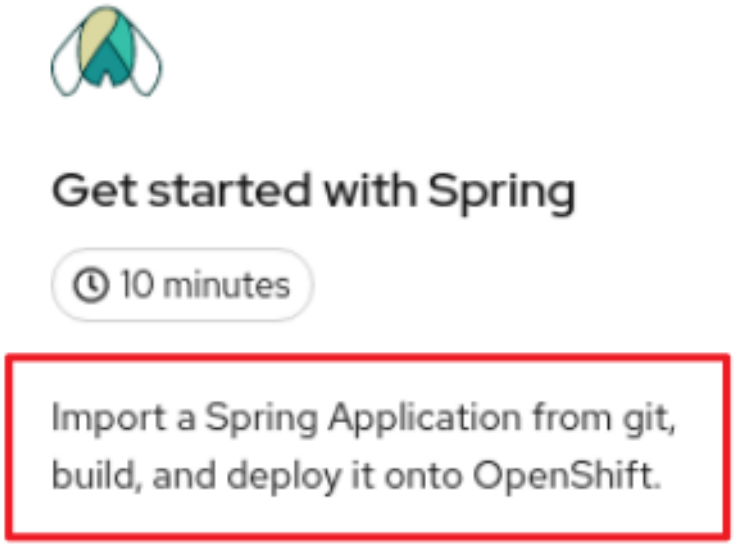
查看 YAML 文件中的 description 元素

```
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 文本

在 web 控制台中查看 description 元素

这个描述会出现在**快速开始**页的介绍中。



10.4.2.3. displayName 元素

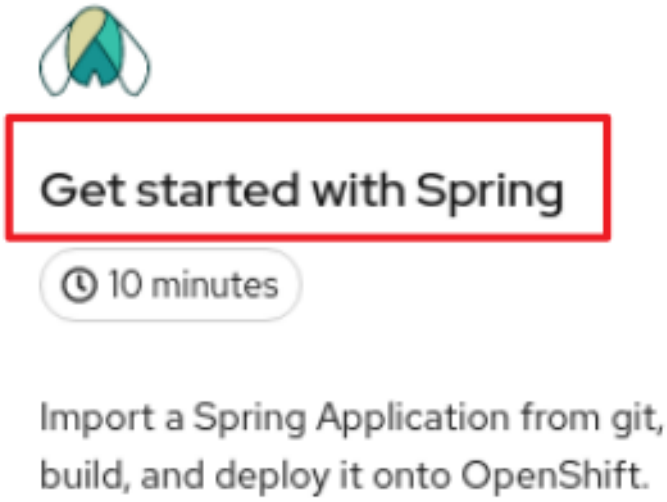
查看 YAML 文件中的 displayName 元素

```
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** 文本。

在 web 控制台中查看 displayName 元素

显示名称会出现在**快速**启动页的介绍中。



10.4.2.4. durationMinutes 元素

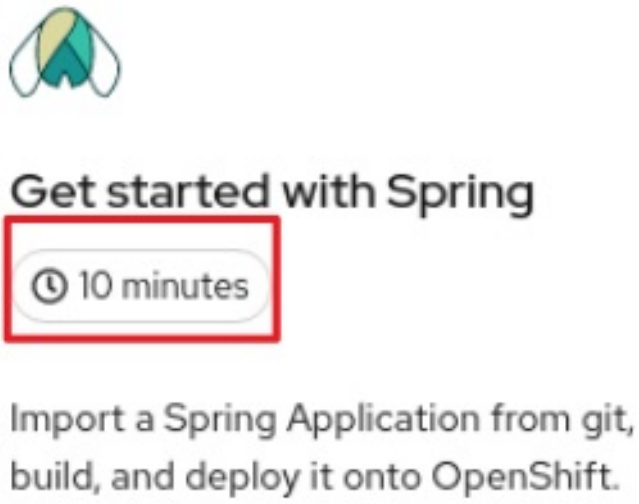
在 YAML 文件中查看 durationMinutes 元素

```
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** 值，以分钟为单位。这个值定义了快速启动完成所需的时间。

在 web 控制台中查看 durationMinutes 元素

durationMinutes 元素会出现在**快速开始**页的介绍中。



10.4.2.5. icon 元素

查看 YAML 文件中的 icon 元素

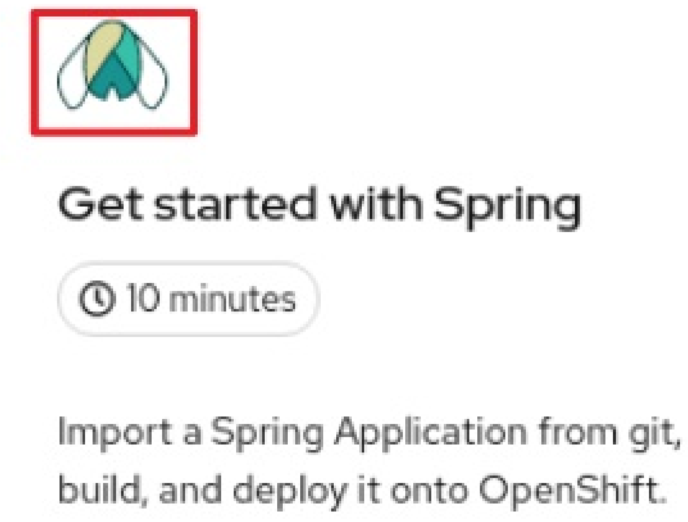

```
...
spec:
  description: 'Import a Spring Application from git, build, and deploy it onto OpenShift.'
  displayName: Get started with Spring
  durationMinutes: 10
  icon: >-   (1)


JMYXllcl8xIiBkYXRhLW5hbWU9IkxheWVyIDEiIHZpZXdCb3g9IjAgMCAxMDI0IDEwMjQiPjxkZWZzPjxzc
HlsZT4uY2xzLTF7ZmlsbDojMTUzZDNjO30uY2xzLTJ7ZmlsbDojZDhkYTlkO30uY2xzLTN7ZmlsbDojNT
hjMGE4O30uY2xzLTR7ZmlsbDojZmZmO30uY2xzLTV7ZmlsbDojM2Q5MTkxO308L3N0eWxlPjwvZGV
mcz48dGl0bGU+c25vd2Ryb3BfaWNvbl9yZ2JfZGVmYXVsdDwvdGl0bGU+PHBhdGggY2xhc3M9ImNsc
y0xIiBkPSJNMTAxMi42OSw1OTNjLTExLjEyLTM4LjA3LTMxLTczLTU5LjIxLTEwMy44LTkuNS0xMS4zL
TIzLjIxLTI4LjI5LTM5LjA2LTQ3Ljk0QzgzMy41MywzNDEsNzQ1LjM3LDIzNC4xOCw2NzQsMTY4Ljk0Yy
01LTUuMjYtMTAuMjYtMTAuMzEtMTUuNjUtMTUuMDdhMjQ2LjQ5LDI0Ni40OSwwLDAsMC0zNi41NS
0yNi44LDE4Mi41LDE4Mi41LDAsMCwwLTIwLjMtMTEuNzcsMjAxLjUzLDIwMS41MywwLDAsMC00My4
xOS0xNUExNTUuMjQsMTU1LjI0LDAsMCwwLDUyOCw5NS4yYy02Ljc2LS42OC0xMS43NC0uODEtM
TQuMzktLjgxaDBsLTEuNjIsMC0xLjYyLDBhMTc3LjMsMTc3LjMsMCwwLDAtMzEuNzcsMy4zNSwyMDg
uMjMsMjA4LjIzLDAsMCwwLTU2LjEyLDE3LjU2LDE4MSwxODEsMCwwLDAtMjAuMjcsMTEuNzUsMjQ
3LjQzLDI0Ny40MywwLDAsMC0zNi41NywyNi44MUMzNjAuMjUsMTU4LjYyLDM1NSwxNjMuNjgsMzUw
LDE2OWMtNzEuMzUsNjUuMjUtMTU5LjUsMTcyLTI0MC4zOSwyNzIuMjhDOTMuNzMsNDYwLjg4LDg
wLDQ3Ny44Nyw3MC41Miw0ODkuMTcsNDIuMzUsNTIwLDI2LjQzLDU1NC45LDExLjMxLDU5MywwLjNz
IsNjI5LjIyLTEuNzMsNjY3LjY5LDQsNzA3LjMxLDE1LDc4Mi40OSw1NS43OCw4NTkuMTIsMTE4LjkzLD
kyMy4wOWEyMiwyMiwwLDAsMCwxNS41OSw2LjUyaDEuODNsMS44Ny0uMzJjODEuMDYtMTMuOT
EsMTEwLTc5LjU3LDE0My40OC0xNTUuNiwzLjkxLTguODgsNy45NS0xOC4wNSwxMi4yLTI3LjQzcTU
uNDIsOC41NCwxMS4zOSwxNi4yM2MzMS44NSw0MC45MSw3NS4xMiw2NC42NywxMzIuMzIsNzIuNj
NsMTguOCwyLjYyLDQuOTUtMTguMzNjNjMuMjYtNDkuMDcsMzUuMy05MC44NSw1MC42NC0xMT
YuMTksMTUuMzQsMjUuMzQsMzcuMzgsNjcuMTIsNTAuNjQsMTE2LjE5bDUsMTguMzMsMTguOC0yL
jYyYzU3LjItOCwxMDAuNDctMzEuNzIsMTMyLjMyLTcyLjYzYTYtNy42OCwxMS4zOS0xNi4yM2M0LjI1L
DkuMzgsOC4yOSwxOC41NSwxMi4yLDI3LjQzLDMzLjQ5LDc2LDYyLjQ2LDE0MS42OSwxNDMuNDgs
MTU1LjZsMS44My4zMWgxLjg5YTIyLDIyLDAsMCwwLDE1LjU5LTYuNTJjNjMuMTUtNjQsMTAzLjk
E0MC42LDExNC44OS0yMTUuNzhDMTAyNS43Myw2NjcuNjksMTAyMy4yOCw2MjkuMjIsMTAxMi42O
Sw1OTNaIi8+PHBhdGggY2xhc3M9ImNscy0yIiBkPSJNMzY0LjE1LDE4NS4yM2MxNy44OS0xNi40LDM
0LjctMzAuMTUsNDkuNzctNDAuMTFhMjEyLDIxMiwwLDAsMSw2NS45My0yNS43M0ExOTgsMTk4LDA
sMCwxLDUxMiwxMTYuMjdhMTk2LjExLDE5Ni4xMSwwLDAsMSwzMiwzLjFjNC41LjkxLDkuMzYsMi4wN
wxNC41MywzLjUyLDYwLjQxLDIwLjQ4LDg0LjkyLDkxLjA1LTQ3LjQ0LDI0OC4wNi0yOC43NSwzNC4x
Mi0xNDAuNywxOTQuODQtMTg0LjY2LDI2OC40MmE2MzAuODYsNjMwLjg2LDAsMCwwLTMzLjIyLD
U4LjMyQzI3Niw2NTUuMzQsMjY1LjQsNTk4LDI2NS40LDUyMC4yOSwyNjUuNCwzNDAuNjEsMzExLjY
5LDI0MC43NCwzNjQuMTUsMTg1LjIzWiIvPjxwYXRoIGNsYXNzPSJjbHMtMyIgZD0iTTUyNy41NCwzO
DQuODNjODQuMDYtOTkuNywxMTYuMDYtMTc3LjI4LDk1LjIyLTIzMC43NCwxMS44Miw2LjY5LDI0LD
E5LjIsMzcuMDYsMzEuMTMsNTIuNDgsNTUuNSw5OC43OCwxNTUuMzgsOTguNzgsMzM1LjA3LDAsNzcuNzEtMTAuNiwxMzUuMDUtMjcuNzcsMTc3LjRhNjI4LjczLDYyOC43MywwLDAsMC0zMy4yMy01OC4zMmMtMzkt
4zMmMtMzktNjUuMjYtMTMxLjQ1LTE5OS0xNzEuOTMtMjUyLjI3QzUyNi4zMywzODYuMjksNTI3LDM4
NS41Miw1MjcuNTQsMzg0LjgzWiIvPjxwYXRoIGNsYXNzPSJjbHMtNCIgZD0iTTEzNC41OCw5MDguM
DdoLS4wNmEuMzkuMzksMCwwLDEtLjI3LS4xMWMtMTE5LjUyLTEyMS4wNy0xNTUtMjg3LjQtNDcuN
TQtNDA0LjU4LDM0LjYzLTQxLjE0LDEyMC0xNTEuNiwyMDIuNzUtMjQyLjE5LTMuMTMsNy02LjEyLDE
0LjI1LTguOTIsMjEuNjktMjQuMzQsNjQuNDUtMzYuNjcsMTQ0LjMyLTM2LjY3LDIzNy40MSwwLDU2LjU
zLDUuNTgsMTA2LDE2LjU5LDE0Ny4xNEEzMDcuNDksMzA3LjQ5LDAsMCwwLDI4MC45MSw3MjND
MjM3LDgxNi44OCwyMTYuOTMsODkzLjgzLDEzNC41OCw5MDguMDdaIi8+PHBhdGggY2xhc3M9ImN
scy01IiBkPSJNNTgzLjQzLDgxMy43OUM1NjAuMTgsNzI3LjcyLDUxMiw2NjQuMTUsNTEyLDY2NC4xN
XMtNDguMTcsNjMuNTctNzEuNDMsMTQ5LjY0Yy00OC40NS02Ljc0LTEwMC45MS0yNy41Mi0xMzUu
NjYtOTEuMThhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Ny03MS41NGwuMjEtLjMyLjE5LS4zM2M2M
```

```
zOC02My42MywxMjYuNC0xOTEuMzcsMTY3LjEyLTI0NS42Niw0MC43MSw1NC4yOCwxMjkuMSwxO
DIsMTY3LjEyLDI0NS42NmwuMTkuMzMuMjEuMzJhNjQ1LjY4LDY0NS42OCwwLDAsMSwzOS41Nyw
3MS41NEM2ODQuMzQsNzg2LjI3LDYzMS44OCw4MDcuMDUsNTgzLjQzLDgxMy43OVoiLz48cGF0a
CBjbGFzcz0iY2xzLTQiIGQ9Ik04ODkuNzUsOTA4YS4zOS4zOSwwLDAsMS0uMjcuMTFoLS4wNkM4M
DcuMDcsODkzLjkzLDc4Nyw4MTYuODgsNzQzLjA5LDcyM2EzMDcuNDksMzA3LjQ5LDAsMCwwLDIwL
jQ1LTU1LjU0YzExLTQxLjExLDE2LjU5LTkwLjYxLDE2LjU5LTE0Ny4xNCwwLTkzLjA4LTEyLjMzLTE3M
y0zNi42Ni0yMzcuNHEtNC4yMi0xMS4xNi04LjkzLTIxLjdjODIuNzUsOTAuNTksMTY4LjEyLDIwMS4wNS
wyMDIuNzUsMjQyLjE5QzEwNDQuNzksNjIwLjU2LDEwMDkuMjcsNzg2Ljg5LDg4OS43NSw5MDhaIi8+
PC9zdmc+Cg==
...
```


1. 定义为 base64 值的图标。

在 web 控制台中查看 icon 元素

这个描述会出现在**快速开始**页的介绍中。



10.4.2.6. introduction 元素

查看 YAML 文件中的 introduction 元素

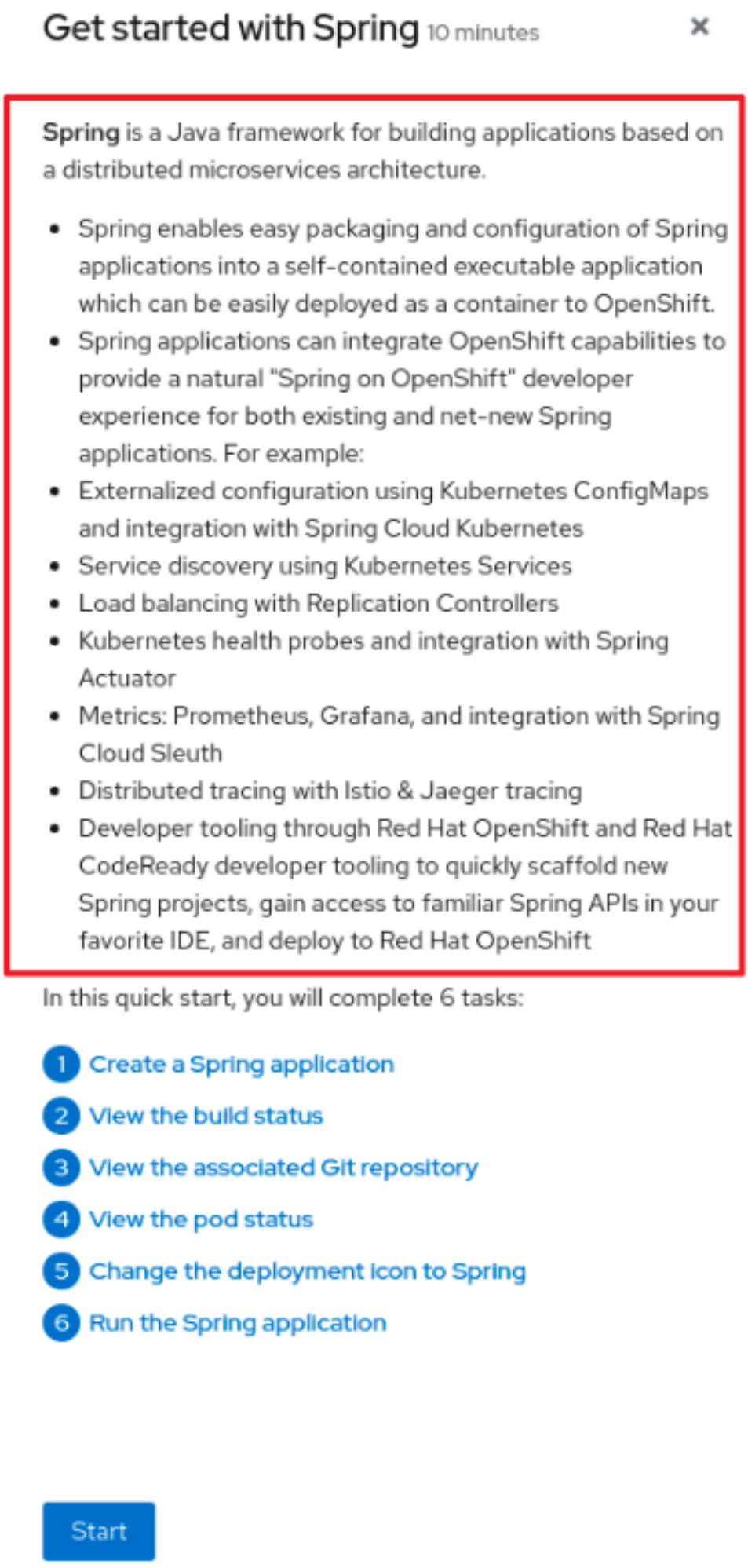
```
...
 introduction: >- (1)
   **Spring** is a Java framework for building applications based on a distributed microservices
architecture.

   - 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 简介介绍了快速启动并列出其中的任务。

在 web 控制台中查看 introduction 元素

点一个快速启动卡后，一个侧边面板滑盘将快速启动并列出它里面的任务。


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

10.4.3. 为快速启动添加自定义图标

为所有快速启动提供了默认图标。您可以提供自己的自定义图标。

流程

1. 查找您要用作自定义图标的 **.svg** 文件。
2. 使用在线工具将文本转换为 base64。
3. 在 YAML 文件中，添加 **icon: >-**，然后在下一行中包含 **data:image/svg+xml;base64**，后面接的是 base64 转换的输出。例如：

```
icon: >-


ciIHJvbGU9ImltZyIgdmlld.
```

10.4.4. 限制对快速开始的访问

并不是所有的快速开始都应该对所有人可用。YAML 文件的 **accessReviewResources** 部分提供限制对快速启动的访问的能力。

只有有权创建 **HelmChartRepository** 资源的用户，才能访问快速开始，使用以下配置：

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

只有用户具有列出 Operator 组和软件包清单（因此能够安装 Operator）时允许用户访问快速开始，使用以下配置：

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

10.4.5. 连接到其他快速开始

流程

- 在 YAML 文件的 **nextQuickStart** 部分，提供您要链接的快速开始的 **name** 而不是 **displayName**。例如：

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

10.4.6. 支持的标签快速启动

使用这些标签在标记中写入快速开始内容。标记将转换为 HTML。

Tag	描述
'b',	粗体文本。
'img',	嵌入图像。
'i',	斜体文本。
'strike',	带有横线贯穿的文本。
's',	较小的文本
'del',	较小的文本。
'em',	加重文本。
'strong',	重要文本。
'a',	anchor 标签。
'p',	段落文本。
'h1',	1 级标题。
'h2',	2 级标题。
'h3',	3 级标题。
'h4',	4 级标题。
'ul',	一个没有顺序的列表。
'ol',	一个有顺序的列表。
'li',	一个列表项。
'code',	代码文本。
'pre',	预格式化的文本块。
'button',	文本中的一个按钮。

10.4.7. 快速入门突出显示 markdown 参考

高亮显示（或提示）功能可让快速入门包含可突出显示并模拟 web 控制台组件的链接。

markdown 语法包含：

- 括起的链接文本
- **highlight** 关键字，后跟您要动画的元素的 ID

10.4.7.1. 视角切换器

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

10.4.7.2. Administrator 视角导航链接

```
[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 视角导航链接

```
[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. 常用导航链接

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

10.4.7.5. Masthea 链接

```
[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. 代码片段 markdown 参考

当 web 控制台的快速入门中包括了一个 CLI 代码片段时，您可以它。要使用这个功能，您必须首先安装 Web Terminal Operator。如果您没有安装 Web Terminal Operator，则 web 终端中执行的 web 终端和代

码片段操作将不存在。另外，无论您是否安装了 Web Terminal Operator，您都可以将代码片段复制到剪贴板中。

10.4.8.1. 内联代码片段的语法

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

注意

如果使用 **execute** 语法，无论您是否安装了 Web Terminal Operator，都会出现 **Copy to clipboard** 操作。

10.4.8.2. 多行代码片段的语法

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

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

10.5. 快速开始内容指南

10.5.1. Card copy

您可以在快速开始卡上自定义标题及描述，但您无法自定义状态。

- 将您的描述长度限制为一到两句。
- 从操作动词开始，并告知用户其目的。正确的示例：

  > Create a serverless application.

10.5.2. 简介

点一个快速启动卡后，一个侧边面板滑盘将快速启动并列出它里面的任务。

- 使您的简介内容更加简洁、明确、易于理解。
- 说明快速开始的目的。用户应在开始之前了解快速开始的目的。
- 为用户提供一个操作，而不是快速开始。
  - **正确的示例**：

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

  - **不正确的示例**：

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

-

- 根据特性的复杂程度，简介应保持在最多四到五句。介绍不要太长。
- 列出简介内容后快速开始的任务，并以操作动词启动每个任务。不要指定任务数量，因为每次添加或删除任务时都需要更新副本。
  - **正确的示例**：

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

  - **不正确的示例**：

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

10.5.3. Task steps

用户点 **Start** 后会出现一系列步骤，它们必须执行这些操作来完成快速开始。

在编写任务步骤时遵循这些常规指南：

- 对于按钮和标签使用 "Click"。在选择框、单选按钮和下拉菜单中使用"选择"。
- 使用 "Click" 而不是 "Click on"
  - **正确的示例**：

    > Click OK.

  - **不正确的示例**：

    > Click on the OK button.

- 告诉用户如何在**管理**员和**开发者**视角间如何切换。即使您认为某个用户可能已经处于正确的视角，最好仍为用户提供相应的操作说明，使其确定位于正确的位置。
  示例：

  > 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.

- 使用 "Location, action" 结构。告诉用户要做什么前先告诉用户到什么地方。
  - **正确的示例**：

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

  - **不正确的示例**：

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

- 保持您的产品术语大写一致。
- 如果您必须指定一个菜单类型或使用列表作为下拉菜单，使用 "dropdown"（一个单词，没有短横线）。
- 明确区分用户动作和产品功能的附加信息。
  - User action：

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

  - Additional information：

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

- 避免方向性的语言，如 "In the top-right corner, click the icon"。因为当 UI 布局改变时，方向语言就有可能变为不正确。另外，桌面用户对于具有不同屏幕大小的用户来说，方向可能是不正确的。反之，使用它的名称来标识项。
  - **正确的示例**：

    > In the navigation menu, click Settings.

  - **不正确的示例**：

    > In the left-hand menu, click Settings.

- 不要只使用颜色来标识项，如 "Click the gray circle"。颜色标识符对受限制的用户，尤其是无法识别颜色的用户可能不可用。相反，使用它的名称或复制来标识项目，如 button copy。
  - **正确的示例**：

    > The success message indicates a connection.

  - **不正确的示例**：

    > The message with a green icon indicates a connection.

- 一致性地使用第二人称（you）：
  - **正确的示例**：

    > Set up your environment.

  - **不正确的示例**：

    > Let's set up our environment.

10.5.4. Check your work module

- 用户完成一个步骤后会出现一个 **Check your work** 模块。这个模块提示用户回答"是"或对步骤结果没有问题，这使得他们有机会复核他们的工作。对于这个模块，您只需要写一个是或不需要问题。
  - 如果用户的回答是 **Yes**，会出现一个标记。
  - 如果用户的回答是 **No**， 会出现一个出错信息，其中包含相关文档的链接。然后，用户可以选择返回并再次进行尝试。

10.5.5. 格式化 UI 元素

使用以下指南格式化 UI 元素：

- 按钮、下拉菜单、标签、字段和其他 UI 控制的副本复制：在 UI 中写入副本并加粗体。
- 所有其他 UI 元素-包括页面、窗口和面板名称：在 UI 中写入该文件并加粗体。
- 代码或用户输入的文本：使用 monospaced 字体。
- 提示：如果包含到导航或 masthead 元素的提示，则使用您链接的文本。
- CLI 命令：使用 monospaced 字体。
- 在运行文本时，在命令中使用粗体 monospaced 字体。
- 如果参数或选项是一个变量值，使用 monospaced 字体。
- 参数使用粗体的 monospaced 字体，选项使用 monospaced 字体。

10.6. 其他资源

- 关于声音和音调的要求，请参考 PatternFly's brand voice and tone guidelines 。
- 关于其他 UX 内容指南，请参考 PatternFly 的 UX 编写风格指南 。