Red Hat JBoss Web Server 6.0

Red Hat JBoss Web Server Operator

为 OpenShift 安装和使用 Red Hat JBoss Web Server Operator 2.0

Last Updated: 2024-06-18

Red Hat JBoss Web Server 6.0 Red Hat JBoss Web Server Operator

为 OpenShift 安装和使用 Red Hat JBoss Web Server Operator 2.0

法律通告

摘要

安装和使用 Red Hat JBoss Web Server Operator 2.0 来管理 Red Hat OpenShift 中的 Web 应用程序

目录

提供有关 RED HAT JBOSS WEB SERVER 文档的反馈 3

使开源包含更多 4

使开源包含更多 5

第 1 章 RED HAT JBOSS WEB SERVER OPERATOR 6

第 2 章 JWS OPERATOR 2.0 发行版中的新内容？ 7
Level-2 Operator 功能 7
为新镜像启用级别-2 无缝集成 7
第 2 级无缝集成，用于重建现有镜像 8
支持 Red Hat JBoss Web Server metering 标签 8
增强的 webImage 参数 9
增强的 webApp 参数 9

第 3 章 从 OPERATORHUB 安装 JWS OPERATOR 11
3.1. 使用 WEB 控制台安装 JWS OPERATOR 11
3.2. 使用命令行安装 JWS OPERATOR 12

第 4 章 部署现有的 JWS 镜像 15

第 5 章 从集群中删除 JWS OPERATOR 17
5.1. 使用 WEB 控制台删除 JWS OPERATOR 17
5.2. 使用命令行删除 JWS OPERATOR 17

第 6 章 为通用或 GITHUB WEBHOOK 创建 SECRET 19

第 7 章 JWS OPERATOR CRD 参数 22
7.1. CRD 参数层次结构 22
7.2. CRD 参数详情 22

提供有关 RED HAT JBOSS WEB SERVER 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 **以创建 ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

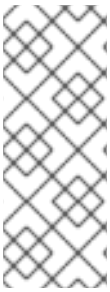
第 1 章 RED HAT JBOSS WEB SERVER OPERATOR

Operator 是一个 Kubernetes 原生应用程序，可以轻松地管理 Kubernetes 和 OpenShift 环境中的复杂有状态应用程序。Red Hat JBoss Web Server (JWS)提供了一个 Operator，用于管理 JWS for OpenShift 镜像。您可以使用 JWS Operator 在 OpenShift 中创建、配置、管理和无缝升级 Web 服务器应用程序实例。

Operator 包括以下关键概念：

- *Operator Framework* 是一个工具包，用于以有效、自动化且可扩展的方式管理 Operator。Operator Framework 由三个主要组件组成：
  - 您可以使用 *OperatorHub* 发现要安装的 Operator。
  - 您可以使用 *Operator Lifecycle Manager* (OLM)在 OpenShift 集群中安装和管理 Operator。
  - 如果要开发自己的自定义 Operator，您可以使用 *Operator SDK*。
- *Operator 组* 是一个 OLM 资源，为 OLM 安装的 Operator 提供多租户配置。Operator 组选择目标命名空间，在其中为部署到与 **OperatorGroup** 对象相同的命名空间中的所有 Operator 生成基于角色的访问控制(RBAC)。
- *自定义资源定义* (CRD)是 Operator 使用的 Kubernetes 扩展机制。CRD 允许 Operator 管理的自定义对象的行为与原生 Kubernetes 对象类似。JWS Operator 提供了一组 CRD 参数，您可以在您要部署的 Web 服务器应用程序的自定义资源文件中指定。

本文档论述了如何安装 JWS Operator、部署现有 JWS 镜像以及从集群中删除 Operator。本文档还详细介绍了 JWS Operator 提供的 CRD 参数。



注意

在按照本文档中的说明进行操作前，您必须确保 OpenShift 集群已安装并根据先决条件进行了配置。有关安装和配置 OpenShift 集群的更多信息，请参阅 OpenShift Container Platform 的安装指南。

有关部署准备的镜像或从现有镜像流构建镜像的更快但更详细指南，请参阅 JWS Operator QuickStart 指南。

重要

红帽支持 JWS 5.4 或更高版本的镜像。

其他资源

- 了解 Operator
- 安装并配置 OpenShift Container Platform 集群

第 2 章 JWS OPERATOR 2.0 发行版中的新内容？

JWS Operator 2.0 发行版本提供 level-2 Operator 功能，如无缝集成。JWS Operator 2.0 还支持 Red Hat JBoss Web Server metering 标签，并包含一些增强的自定义资源定义(CRD)参数。

Level-2 Operator 功能

JWS Operator 2.0 提供以下 level-2 Operator 功能：

- 启用无缝升级
- 支持补丁和次版本升级
- 管理 JWS Operator 1.1.x 部署的 Web 服务器。

为新镜像启用级别-2 无缝集成

DeploymentConfig 对象定义包含一个触发器，供 OpenShift 在新镜像推送到镜像流时部署新容器集。镜像流可以监控新镜像的存储库，或者您可以指示新镜像可供使用的镜像流。

流程

1. 在项目命名空间中，使用 **oc import-image** 命令导入镜像的标签和其他信息来创建镜像流。
例如：

```
oc import-image <my-image>-imagestream:latest \
--from=quay.io/$user/<my-image>:latest \
--confirm
```

在前面的示例中，将出现的每个 & ***lt;my-image*** > 替换为您要导入的镜像的名称。

以上命令通过为 **quay.io/$user/*<my-image>*** 镜像导入信息来创建一个名为 ***<my-image>*-imagestream** 的镜像流。有关镜像流的格式和管理的更多信息，请参阅管理镜像流。

2. 为您希望在镜像流被更新时 JWS Operator 进行部署的 Web 应用程序创建 **WebServer** kind 的自定义资源。您可以使用 YAML 文件格式定义自定义资源。
例如：

```
apiVersion: web.servers.org/v1alpha1
kind: WebServer
metadata:
  name: <my-image>
spec:
  # Add fields here
  applicationName: my-app
  useSessionClustering: true
  replicas: 2
  webImageStream:
    imageStreamNamespace: <project-name>
    imageStreamName: <my-image>-imagestream
```

3. 使用 **oc tag** 命令触发对镜像流的更新。
例如：

```
oc tag quay.io/$user/<my-image> <my-image>-imagestream:latest --scheduled
```

以上命令会导致 OpenShift Container Platform 定期更新指定的镜像流标签。此周期是集群范围的设置，默认设置为 15 分钟。

第 2 级无缝集成，用于重建现有镜像

BuildConfig 对象定义包含用于镜像流更新和 Webhook 的触发器，后者是 GitHub 或通用 Webhook，允许在 Git 或 GitHub 触发 webhook 时重新构建镜像。

有关为 webhook 创建 secret 的更多信息，请参阅为 通用或 GitHub Webhook 创建 secret 。

有关在自定义资源 WebServer 文件中配置通用或 GitHub Webhook 的更多信息，请参阅 JWS Operator CRD 参数。

支持 Red Hat JBoss Web Server metering 标签

JWS Operator 2.0 支持在 JWS Operator 创建的 Red Hat JBoss Web Server pod 中添加 metering 标签。

Red Hat JBoss Web Server 可以使用以下 metering 标签：

- **com.company: Red_Hat**
- **rht.prod_name: Red_Hat_Runtimes**
- **rht.prod_ver: 2023-Q4**
- **rht.comp: JBoss_Web_Server**
- **rht.comp_ver: 6.0.0**
- **rht.subcomp: Tomcat 10**
- **rht.subcomp_t: application**

您可以在您要部署的 Web 应用程序的自定义资源 **WebServer** 文件的 **metadata** 部分下添加标签。例如：

```
---
apiVersion: web.servers.org/v1alpha1
kind: WebServer
metadata:
  name: <my-image>
  labels:
    com.company: Red_Hat
    rht.prod_name: Red_Hat_Runtimes
    rht.prod_ver: 2023-Q4
    rht.comp: JBoss_Web_Server
    rht.comp_ver: 6.0.0
    rht.subcomp: Tomcat 10
    rht.subcomp_t: application
spec:
----
```

注意

如果您为部署的 Web 服务器更改任何标签键或标签值，JWS Operator 会重新部署 Web 服务器应用程序。如果部署的 Web 服务器是从源代码构建的，JWS Operator 也会重建 Web 服务器应用程序。

增强的 **webImage** 参数

在 JWS Operator 2.0 发行版本中，CRD 中的 **webImage** 参数包含以下附加字段：

- **imagePullSecret**
  JWS Operator 用来从存储库中拉取镜像的 secret

注意

secret 必须包含键 **.dockerconfigjson**。JWS Operator 挂载并使用 secret （例如 **--authfile /mount_point/.dockerconfigjson**）从存储库中拉取镜像。**Secret** 对象定义文件可能包含服务器用户名和密码值或令牌，以允许访问由 JWS Operator 构建的镜像流、构建器镜像和镜像中的镜像。

- **webApp**
  一组参数，用于描述 JWS Operator 如何构建 Web 服务器应用程序

增强的 **webApp** 参数

在 JWS Operator 2.0 发行版本中，CRD 中的 **webApp** 参数包含以下附加字段：

- **name**
  Web 服务器应用程序的名称
- **sourceRepositoryURL**
  应用程序源文件所在的 URL
- **sourceRepositoryRef**
  Operator 使用的源存储库的分支
- **sourceRepositoryContextDir**
  pom.xml 文件所在的子目录，必须运行 **mvn install** 命令的位置
- **webAppWarImage**
  JWS Operator 推送构建镜像的镜像的 URL
- **webAppWarImagePushSecret**
  JWS Operator 用来将镜像推送到存储库的 secret
- **builder**
  包含构建 Web 应用所需的所有信息的一组参数，并将镜像创建并推送到镜像存储库

注意

为确保构建器能够成功运行并使用不同的用户 ID 运行命令，构建器必须有权访问 **anyuid** 安全性上下文约束(SCC)。

要授予构建器对 **anyuid** SCC 的访问权限，请输入以下命令：

oc adm policy add-scc-to-user anyuid -z builder

builder 参数包含以下字段：

  - **image**
    构建 Web 应用程序的容器的镜像（例如： **quay.io/$user/tomcat10-buildah**）

- **imagePullSecret**
  JWS Operator 用来从存储库中拉取构建器镜像的 secret （如果指定）

- **applicationBuildScript**
  构建器镜像用于构建应用 **.war** 文件的脚本，并将它移到 **/mnt** 目录

注意

如果没有为此参数指定值，构建器镜像将使用使用 Maven 和 Buildah 的默认脚本。

其他资源

- Operator 能力级别

第 3 章 从 OPERATORHUB 安装 JWS OPERATOR

您可以从 OperatorHub 安装 JWS Operator，以便在 OpenShift 集群中部署和管理 JBoss Web Server 应用程序。OperatorHub 是 Operator Framework 的一个组件，可用于发现您要安装的 Operator。OperatorHub 与 Operator Lifecycle Manger (OLM)协同工作，后者在集群中安装和管理 Operator。

您可以使用以下方法之一从 OperatorHub 安装 JWS Operator：

- 使用 OpenShift Web 控制台。
- 使用 **oc** 命令行工具。

3.1. 使用 WEB 控制台安装 JWS OPERATOR

如果要使用图形用户界面安装 JWS Operator，您可以使用 OpenShift Web 控制台安装 JWS Operator。

注意

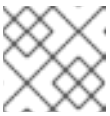
当使用 Web 控制台安装 JWS Operator 时，Operator 使用 **SingleNamespace** 安装模式，Operator 会自动安装 **OperatorGroup** 和 **Subscription** 对象。

先决条件

- 已使用具有集群管理员和 Operator 安装权限的账户部署了 OpenShift Container Platform 集群。

流程

1. 打开 Web 控制台并选择 **Operators > OperatorHub**。
2. 在 **Filter by keyword** 搜索字段中，键入 "JWS"。
3. 选择 JWS Operator。
4. 在 **JBoss Web Server Operator** 菜单中，选择要使用的 **Capability 级别**，然后点 **Install**。
5. 在 **Install Operator** 页面中，执行以下步骤：
   a. 选择提供了 JWS Operator 的 **Update 频道**。



注意

JWS Operator 目前仅通过一个频道获得。

   b. **选择** Operator 的安装模式。
   您可以将 Operator 安装到所有命名空间或集群中的特定命名空间。如果选择特定的命名空间选项，请使用 **Installed Namespace** 字段指定要安装 Operator 的命名空间。

注意

如果没有指定命名空间，Operator 会默认安装到集群中的所有命名空间。

   c. 为 Operator 选择 **批准策略**。
   请考虑以下指南：

- 如果选择 **Automatic** 更新，当有新版本的 Operator 可用时，OLM 会自动升级 Operator 的运行实例。
- **如果选择手动** 更新，则当有新版 Operator 可用时，OLM 会创建更新请求。作为集群管理员，您必须手动批准更新请求，以确保 Operator 更新至新版本。

6. 点 **Install**。

注意

如果您选择了一个 **Manual** 批准策略，则必须在安装完成前批准安装计划。

然后 JWS Operator 会出现在 **Operators** 选项卡的 **Installed Operators** 部分。

3.2. 使用命令行安装 JWS OPERATOR

如果要使用命令行界面安装 JWS Operator，您可以使用 **oc** 命令行工具安装 JWS Operator。红帽提供的 JWS Operator 被命名为 **jws-operator**。

从命令行安装 JWS Operator 的步骤包括验证 Operator 支持的安装模式和可用频道以及创建 Subscription 对象。根据 Operator 使用的安装模式，您可能需要在创建 Subscription 对象前在项目命名空间中创建 Operator 组。

先决条件

- 已使用具有 Operator 安装权限的账户部署了 OpenShift Container Platform 集群。
- 您已在本地系统上安装了 **oc** 工具。

流程

1. 要检查 JWS Operator，请执行以下步骤：
   a. 查看 OperatorHub 中集群可用的 JWS Operator 列表：

```
$ oc get packagemanifests -n openshift-marketplace | grep jws
```

   前面的命令显示每个可用 Operator 的名称、目录和年龄。

   例如：

```
NAME           CATALOG           AGE
jws-operator   Red Hat Operators   16h
```

   b. 检查 JWS Operator，以验证 Operator 支持的安装模式和可用频道：

```
$ oc describe packagemanifests jws-operator -n openshift-marketplace
```

2. 检查 Operator 组的实际列表：

```
$ oc get operatorgroups -n <project_name>
```

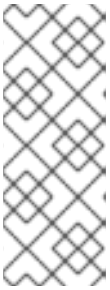
在前面的示例中，将 &lt ***;project_name&*** gt; 替换为您的 OpenShift 项目名称。

前面的命令显示每个可用 Operator 组的名称和年龄。

例如：

```
NAME      AGE
mygroup   17h
```

3. 如果需要创建 Operator 组，请执行以下步骤：



注意

如果要安装的 Operator 使用 **SingleNamespace** 安装模式，且您还没有适当的 Operator 组，则必须完成这个步骤来创建 Operator 组。您必须确保在指定命名空间中只创建一个 Operator 组。

如果要安装的 Operator 使用 **AllNamespaces** 安装模式，或者已有适当的 Operator 组，您可以忽略这一步。

   a. 为 **OperatorGroup** 对象创建 YAML 文件。
   例如：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <project_name>
spec:
  targetNamespaces:
  - <project_name>
```

   在前面的示例中，将 ***<operatorgroup_name>*** 替换为您要创建的 Operator 组的名称，并将 ***lt;project_name>*** 替换为您要安装 Operator 的项目的名称。要查看项目名称，您可以运行 **oc project -q** 命令。

   b. 从 YAML 文件创建 **OperatorGroup** 对象：

```
$ oc apply -f <filename>.yaml
```

   在前面的示例中，将 ***<filename>*** **.yaml** 替换为您为 **OperatorGroup** 对象创建的 YAML 文件的名称。

4. 要创建 Subscription 对象，请执行以下步骤：

   a. 为 **Subscription** 对象创建一个 YAML 文件。
   例如：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
   name: jws-operator
   namespace: <project_name>
spec:
   channel: alpha
```

```
name: jws-operator
source: redhat-operators
sourceNamespace: openshift-marketplace
```

在前面的示例中，将 ***<project_name>*** 替换为您要安装 Operator 的项目的名称。要查看项目名称，您可以运行 **oc project -q** 命令。

您指定的命名空间必须具有与 Operator 相同的安装模式设置的 **OperatorGroup** 对象。如果 Operator 使用 **AllNamespaces** 安装模式，请将 &lt ***;project_name&*** gt; 替换为 **openshift-operators**，该模式已提供适当的 Operator 组。如果 Operator 使用 **SingleNamespace** 安装模式，请确保此命名空间只有一个 **OperatorGroup** 对象。

确保 **source** 设置与验证 Operator 可用的频道时显示的 **Catalog Source** 值匹配（如 **redhat-operators**）。

b. 从 YAML 文件创建 **Subscription** 对象：

```
$ oc apply -f <filename>.yaml
```

在前面的示例中，将 ***<filename>*** **.yaml** 替换为您为 **Subscription** 对象创建的 YAML 文件的名称。

验证

- 要验证 JWS Operator 是否已成功安装，请输入以下命令：

```
$ oc get csv -n <project_name>
```

在前面的示例中，将 & ***lt;project_name*** > 替换为安装 Operator 的项目的名称。

前面的命令显示已安装的 Operator 的详细信息。

例如：

NAME	显示	版本	替换	阶段
jws-operator.v2.0.*x*	JWS Operator	2.0.*x*	jws-operator.v2.0.*y*	Succeeded

在前面的输出中，**2.0.*x*** 代表当前的 Operator 版本（如 **2.0.6**），而 **2.0.*y*** 代表当前版本替换的以前的 Operator 版本（如 **2.0.5**）。

第 4 章 部署现有的 JWS 镜像

您可以使用 JWS Operator 促进您要部署到 OpenShift 集群中的 Web 服务器应用程序的现有镜像部署。在这种情况下，您必须为要部署的 Web 服务器应用程序创建一个自定义资源 **WebServer** 文件。JWS Operator 使用自定义资源 **WebServer** 文件来处理应用程序部署。

先决条件

- 您已从 OperatorHub 安装了 JWS Operator。
  要确保安装了 JWS Operator，请输入以下命令：

```
$ oc get deployment.apps/jws-operator-controller-manager
```

前面的命令显示 Operator 的名称和状态详情。

例如：

```
NAME          READY  UP-TO-DATE  AVAILABLE  AGE
jws-operator  1/1    1           1          15h
```

注意

如果要查看更详细的输出，您可以使用以下命令：

oc describe deployment.apps/jws-operator-controller-manager

流程

1. 准备镜像并将其推送到要显示镜像的位置（例如： **quay.io/ <*USERNAME*> /tomcat-demo:latest**）。
2. 要为 Web 服务器应用程序创建自定义资源文件，请执行以下步骤：
   a. 创建名为 的 YAML 文件，如 **webservers_cr.yaml**。
   b. 以以下格式输入详情：

```
apiVersion: web.servers.org/v1alpha1
kind: WebServer
metadata:
   name: <image name>
spec:
   # Add fields here
   applicationName: <application name>
   replicas: 2
webImage:
  applicationImage: <URL of the image>
```

例如：

```
apiVersion: web.servers.org/v1alpha1
kind: WebServer
metadata:
   name: example-image-webserver
```

```
spec:
   # Add fields here
   applicationName: jws-app
   replicas: 2
webImage:
  applicationImage: quay.io/<USERNAME>/tomcat-demo:latest
```

3. 要部署 Web 应用程序，请执行以下步骤：

   a. 前往您创建 Web 应用程序的目录。

   b. 输入以下命令：

   ```
   $ oc apply -f webservers_cr.yaml
   ```

   前面的命令会显示一条消息，以确认 Web 应用已经部署。

   例如：

   ```
   webserver/example-image-webserver created
   ```

   运行上述命令时，Operator 还会自动创建路由。

4. 验证 Operator 自动创建的路由：

   ```
   $ oc get routes
   ```

5. 可选：删除在第 3 步中创建的 **webserver** ：

   ```
   $ oc delete webserver example-image-webserver
   ```

注意

或者，您可以通过删除 YAML 文件来删除 **webserver**。例如：

oc delete -f webservers_cr.yaml

其他资源

- 路由配置：创建基于 HTTP 的路由

第 5 章 从集群中删除 JWS OPERATOR

如果您不再需要使用 JWS Operator，您可以从集群中删除 JWS Operator。

您可以使用以下方法之一从集群中删除 JWS Operator：

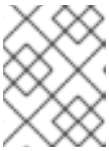
- 使用 OpenShift Web 控制台。
- 使用 **oc** 命令行工具。

5.1. 使用 WEB 控制台删除 JWS OPERATOR

如果要使用图形用户界面删除 JWS Operator，您可以使用 OpenShift Web 控制台删除 JWS Operator。

先决条件

- 已使用具有集群管理员权限的帐户部署了 OpenShift Container Platform **集群**。



注意

如果您没有 **集群管理员权限**，可以绕过这个要求。如需更多信息，请参阅允许非集群管理员安装 Operator。

流程

1. 打开 web 控制台并点 **Operators > Installed Operators**。
2. 选择 **Actions** 菜单并点击 **Uninstall Operator**。

注意

Uninstall Operator 选项会自动删除 Operator、任何 Operator 部署和 Pod。

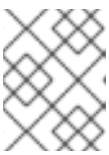
删除 Operator 不会删除 Operator 的任何自定义资源定义或自定义资源，包括 CRD 或 CR。如果 Operator 在集群中部署了应用程序，或者 Operator 在集群外配置了资源，则必须手动清理这些应用程序和资源。

5.2. 使用命令行删除 JWS OPERATOR

如果要使用命令行界面删除 JWS Operator，您可以使用 **oc** 命令行工具删除 JWS Operator。

先决条件

- 已使用具有集群管理员权限的帐户部署了 OpenShift Container Platform **集群**。



注意

如果您没有 **集群管理员权限**，可以绕过这个要求。如需更多信息，请参阅允许非集群管理员安装 Operator。

- 您已在本地系统上安装了 **oc** 工具。

流程

1. 检查订阅的 Operator 的当前版本：

```
$ oc get subscription jws-operator -n <project_name> -o yaml | grep currentCSV
```

在前面的示例中，将 &***lt;project_name*** > 替换为安装 Operator 的项目的命名空间。如果 Operator 已安装到所有命名空间，请将 <***project_name>*** 替换为 **openshift-operators**。

前面的命令显示以下输出，其中 **v2.0.*x*** 代表 Operator 版本（如 **v2.0.6**）：

```
f:currentCSV: {}
currentCSV: jws-operator.v2.0.x
```

2. 删除 Operator 的订阅：

```
$ oc delete subscription jws-operator -n <project_name>
```

在前面的示例中，将 &***lt;project_name*** > 替换为安装 Operator 的项目的命名空间。如果您的 Operator 已安装到所有命名空间，请将 <***project_name>*** 替换为 **openshift-operators**。

3. 删除目标命名空间中 Operator 的 CSV：

```
$ oc delete clusterserviceversion <currentCSV> -n <project_name>
```

在前面的示例中，将 &lt ***;currentCSV*** > 替换为在第 1 步中 获取的 **currentCSV** 值（例如，**jws-operator.v2.0.6**）。将 ***<project_name*** > 替换为安装 Operator 的项目的命名空间。如果您的 Operator 已安装到所有命名空间，请将 <***project_name>*** 替换为 **openshift-operators**。

前面的命令会显示一条消息，以确认 CSV 已被删除。

例如：

```
clusterserviceversion.operators.coreos.com "jws-operator.v2.0.x" deleted
```

第 6 章 为通用或 GITHUB WEBHOOK 创建 SECRET

您可以创建一个可与通用或 GitHub Webhook 搭配使用的 secret，以触发 Git 存储库中的应用构建。根据用于应用程序代码的 Git 托管平台类型，JWS Operator 提供了一个 **genericWebhookSecret** 参数和一个 **githubWebhookSecret** 参数，可用于在 web 应用程序的自定义资源文件中指定 secret。

流程

1. 创建编码的 secret 字符串：
   a. 创建名为 的文件，如 **secret.txt**。
   b. 在 **secret.txt** 文件中，以纯文本形式输入 secret 字符串。
   例如：

   ```
   qwerty
   ```

   c. 要对字符串进行编码，请输入以下命令：

   ```
   base64 secret.txt
   ```

   前面的命令显示编码的字符串。

   例如：

   ```
   cXdlcnR5Cg==
   ```

2. 创建定义 **Secret** 对象的 **secret.yaml** 文件。
   例如：

   ```
   kind: Secret
   apiVersion: v1
   metadata:
     name: jws-secret
   data:
     WebHookSecretKey: cXdlcnR5Cg==
   ```

   在前面的示例中，**jws-secret** 是 secret 的名称， **cXdlcnR5Cg==** 是编码的 secret 字符串。

3. 运行以下命令来创建 secret：

   ```
   oc create -f secret.yaml
   ```

   前面的命令会显示一条消息，以确认该机密已创建好。

   例如：

   ```
   secret/jws-secret created
   ```

验证

1. 获取 Webhook 的 URL：

```
oc describe BuildConfig | grep webhooks
```

前面的命令生成 Webhook URL，格式为：

```
https://<host>:
<port>/apis/build.openshift.io/v1/namespaces/<namespace>/buildconfigs/<name>/webhooks/
<secret>/generic
```

2. 创建名为 的最小 JSON 文件，如 **payload.json** ：

```
{}
```

3. 要向 webhook 发送请求，请输入以下 **curl** 命令：

```
curl -H "X-GitHub-Event: push" -H "Content-Type: application/json" -k -X POST --data-binary
@payload.json https://<host>:
<port>/apis/build.openshift.io/v1/namespaces/<namespace>/buildconfigs/<name>/webhooks/
<secret>/generic
```

在前面的示例中，**payload.json** 是您创建的最小 JSON 文件的名称。

将 URL 字符串中的 & ***lt;host > , <port > , <namespace > , 和 <name*** > 替换为适合您环境的值。将 ***<secret*** > 替换为您为 Webhook 创建的 secret 的名称。

前面的命令以 JSON 格式生成以下类型的 Webhook 响应：

```
{"kind":"Build","apiVersion":"build.openshift.io/v1","metadata":{"name":"test-
2","namespace":"jfc","selfLink":"/apis/build.openshift.io/v1/namespaces/jfc/buildconfigs/test-
2/instantiate","uid":"a72dd529-edc6-4e1c-898e-
7c0dbbea176e","resourceVersion":"846159","creationTimestamp":"2020-10-
30T12:29:30Z","labels":{"application":"test","buildconfig":"test","openshift.io/build-
config.name":"test","openshift.io/build.start-policy":"Serial"},"annotations":{"openshift.io/build-
config.name":"test","openshift.io/build.number":"2"},"ownerReferences":
[{"apiVersion":"build.openshift.io/v1","kind":"BuildConfig","name":"test","uid":"1f78fa3f-2f3b-
421b-9f49-192184cc2280","controller":true}],"managedFields":[{"manager":"openshift-
apiserver","operation":"Update","apiVersion":"build.openshift.io/v1","time":"2020-10-
30T12:29:30Z","fieldsType":"FieldsV1","fieldsV1":{"f:metadata":{"f:annotations":{".":
{},"f:openshift.io/build-config.name":{},"f:openshift.io/build.number":{}},"f:labels":{".":
{},"f:application":{},"f:buildconfig":{},"f:openshift.io/build-config.name":
{},"f:openshift.io/build.start-policy":{}},"f:ownerReferences":{".":{},"k:{\"uid\":\"1f78fa3f-2f3b-
421b-9f49-192184cc2280\"}":{".":{},"f:apiVersion":{},"f:controller":{},"f:kind":{},"f:name":
{},"f:uid":{}}}},"f:spec":{"f:output":{"f:to":{".":{},"f:kind":{},"f:name":{}}},"f:serviceAccount":
{},"f:source":{"f:contextDir":{},"f:git":{".":{},"f:ref":{},"f:uri":{}},"f:type":{}},"f:strategy":
{"f:sourceStrategy":{".":{},"f:env":{},"f:forcePull":{},"f:from":{".":{},"f:kind":{},"f:name":
{}},"f:pullSecret":{".":{},"f:name":{}}},"f:type":{}},"f:triggeredBy":{}},"f:status":{"f:conditions":{".":
{},"k:{\"type\":\"New\"}":{".":{},"f:lastTransitionTime":{},"f:lastUpdateTime":{},"f:status":
{},"f:type":{}}},"f:config":{".":{},"f:kind":{},"f:name":{},"f:namespace":{}},"f:phase":{}}}}]},"spec":
{"serviceAccount":"builder","source":{"type":"Git","git":{"uri":"https://github.com/jfclere/demo-
webapp.git","ref":"master"},"contextDir":"/"},"strategy":{"type":"Source","sourceStrategy":
{"from":{"kind":"DockerImage","name":"image-registry.openshift-image-
registry.svc:5000/jfc/jboss-webserver54-tomcat9-
openshift@sha256:75dcdf81011e113b8c8d0a40af32dc705851243baa13b68352706154174319e7"},"pullSecret":{"name":"builder-dockercfg-rvbh8"},"env":
[{"name":"MAVEN_MIRROR_URL"},{"name":"ARTIFACT_DIR"}],"forcePull":true}},"output":
```

{"to":{"kind":"ImageStreamTag","name":"test:latest"}},"resources":{},"postCommit":
{},"nodeSelector":null,"triggeredBy":[{"message":"Generic WebHook","genericWebHook":
{"secret":"\u003csecret\u003e"}}]},"status":{"phase":"New","config":
{"kind":"BuildConfig","namespace":"jfc","name":"test"},"output":{},"conditions":
[{"type":"New","status":"True","lastUpdateTime":"2020-10-
30T12:29:30Z","lastTransitionTime":"2020-10-30T12:29:30Z"}]}}

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Success",
  "message": "no git information found in payload, ignoring and continuing with build",
  "code": 200
}
```

其他资源

- Webhook 触发器
- JWS Operator CRD 参数

第 7 章 JWS OPERATOR CRD 参数

JWS Operator 提供了一组自定义资源定义(CRD)参数。为 web 应用程序创建自定义资源 **WebServer** 文件时，您可以使用 < ***key> : < value> 格式指定*** 参数值。JWS Operator 使用您在自定义资源 **WebServer** 文件中指定的信息来部署 Web 应用程序。

7.1. CRD 参数层次结构

JWS Operator 以以下分级格式提供 CRD 参数：

```
applicationName: <value>
replicas: <value>
useSessionClustering: <value>
webImage:
  applicationImage: <value>
  imagePullSecret: <value>
  webApp:
    name: <value>
    sourceRepositoryURL: <value>
    sourceRepositoryRef: <value>
    contextDir: <value>
    webAppWarImage: <value>
    webAppWarImagePushSecret: <value>
    builder:
      image: <value>
      imagePullSecret: <value>
      applicationBuildScript: <value>
  webServerHealthCheck:
    serverReadinessScript: <value>
    serverLivenessScript: <value>
webImageStream:
  imageStreamName: <value>
  imageStreamNamespace: <value>
  webSources:
    sourceRepositoryUrl: <value>
    sourceRepositoryRef: <value>
    contextDir: <value>
    webSourcesParams:
      mavenMirrorUrl: <value>
      artifactDir: <value>
      genericWebHookSecret: <value>
      githubWebHookSecret: <value>
  webServerHealthCheck:
    serverReadinessScript: <value>
    serverLivenessScript: <value>
```

注意

当您创建自定义资源 **WebServer** 文件时，以与上例概述相同的分级格式指定参数名称和值。有关创建自定义资源 **WebServer** 文件的更多信息，请参阅部署现有的 JWS 镜像。

7.2. CRD 参数详情

下表描述了 JWS Operator 提供的 CRD 参数。下表显示了在层次结构中超过它的任何更高级别参数上下文中的每个参数名称。

参数名称	描述
replicas	要运行的 JBoss Web 服务器镜像的 pod 数量 例如： **replicas: 2**
applicationName	您希望 JWS Operator 部署的 Web 应用程序名称 应用名称必须是 OpenShift 命名空间或项目中的唯一值。JWS Operator 使用您指定的应用程序名称来创建路由来访问 Web 应用程序。 例如： **applicationName: my-app**
useSessionClustering	启用 DNSping 会话集群 默认设置为 **false**。如果将此参数设置为 **true**，则镜像必须基于 JBoss Web Server 镜像，因为会话集群使用 **ENV_FILES** 环境变量和一个 shell 脚本在 **server.xml** 文件中添加集群。  **注意** 在本发行版本中，会话集群功能仅作为技术预览功能提供。当前 Operator 版本使用 **DNS Membership 提供者**，这受到 DNS 限制的限制。**InetAddress.getAllByName( )** 结果会被缓存，这意味着扩展时会话复制可能无法正常工作。 例如： **useSessionClustering: true**
webImage	控制 JWS Operator 如何从现有镜像部署 pod 的一组参数 此参数包含 **applicationImage**,**imagePullSecret**,**webApp**, 和 **webServerHealthCheck** 字段。
WebImage: applicationImage	要部署的应用程序镜像名称的完整路径 例如： **applicationImage: quay.io/$user/my-image-name**

参数名称	描述
webImage: **imagePullSecret**	JWS Operator 用来从存储库中拉取镜像的 secret 名称 secret 必须包含键 **.dockerconfigjson**。JWS Operator 挂载 secret，并使用类似于 **--authfile /mount_point/.dockerconfigjson** 从存储库中拉取镜像。 **Secret** 对象定义文件可能包含多个用户名和密码值或令牌，以允许访问由 JWS Operator 构建的镜像流、构建器镜像和镜像中的镜像。 例如： **imagePullSecret: mysecret**
WebImage: **webApp**	一组参数，用于描述 JWS Operator 如何构建您要添加到应用程序镜像的 Web 应用程序 如果没有指定 **webApp** 参数，JWS Operator 会在不构建应用程序的情况下部署 Web 应用程序。 此参数包含 **名称**,**sourceRepositoryURL**,**sourceRepositoryRef**,**contextDir**,**webAppWarImage**,**webAppWarImagePushSecret**, 和 **builder** 字段。
WebImage: **webApp:** **名称**	Web 应用程序文件的名称 默认名称为 **ROOT.war**。 例如： **name: my-app.war**
WebImage: **webApp:** **sourceRepositoryURL**	应用程序源文件所在的 URL 源应包含 Maven **pom.xml** 文件，以支持 Maven 构建。当 Maven 为应用程序生成 **.war** 文件时，**.war** 文件将复制到 JWS Operator 用于部署应用程序的镜像的 **webapps** 目录中（例如 **/opt/jws-5.x/tomcat/webapps**）。 例如： **sourceRepositoryUrl: 'https://github.com/$user/demo-webapp.git'**
WebImage: **webApp:** **sourceRepositoryRef**	JWS Operator 使用的源存储库的分支 例如： **sourceRepositoryRef: main**

参数名称	描述
WebImage: **webApp:** **contextDir**	运行 **pom.xml** 文件所在的源存储库中的子目录，并且运行 **mvn install** 命令 例如： **contextDir: /**
WebImage: **webApp:** **webAppWarImage**	JWS Operator 推送构建镜像的镜像的 URL
WebImage: **webApp:** **webAppWarImagePushSecret**	JWS Operator 用来将镜像推送到存储库的 secret 名称 secret 必须包含键 **.dockerconfigjson**。JWS Operator 挂载 secret，并使用类似于 **--authfile /mount_point/.dockerconfigjson** 来将镜像推送到存储库。 如果 JWS Operator 使用 pull secret 从存储库中拉取镜像，您必须将 pull secret 的名称指定为 **webAppWarImagePushSecret** 参数的值。如需更多信息，请参阅 imagePullSecret。 例如： **imagePullSecret: mysecret**
WebImage: **webApp:** **builder**	一组参数，用于描述 JWS Operator 如何构建 Web 应用程序并将镜像推送到镜像存储库  **注意** 为确保构建器能够成功运行并使用不同的用户 ID 运行命令，构建器必须有权访问 **anyuid** SCC （安全上下文约束）。要授予构建器对 **anyuid** SCC 的访问权限，请输入以下命令： **oc adm policy add-scc-to-user anyuid -z builder** 此参数包含 **镜像**、**imagePullSecret** 和 **applicationBuildScript** 字段。
WebImage: **webApp:** **builder:** **image**	JWS Operator 构建 Web 应用程序的容器的镜像 例如： **image: quay.io/$user/tomcat10-buildah**

参数名称	描述
WebImage: **webApp:** **builder:** **imagePullSecret**	JWS Operator 用来从存储库中拉取构建器镜像的 secret 名称（如果指定） secret 必须包含键 **.dockerconfigjson**。JWS Operator 挂载 secret，并使用类似于 **--authfile /mount_point/.dockerconfigjson** 从存储库中拉取镜像。 **Secret** 对象定义文件可能包含多个用户名和密码值或令牌，以允许访问由 JWS Operator 构建的镜像流、构建器镜像和镜像中的镜像。 例如： **imagePullSecret: mysecret**
WebImage: **webApp:** **builder:** **applicationBuildScript**	构建器镜像用于构建应用 **.war** 文件的脚本，并将它移到 **/mnt** 目录 如果没有为此参数指定值，构建器镜像将使用使用 Maven 和 Buildah 的默认脚本。
webImage: **webServerHealthCheck**	JWS Operator 使用的健康检查 默认行为是使用健康 valve，它不需要任何参数。 此参数包含 **serverReadinessScript** 和 **serverLivenessScript** 字段。
WebImage: **webServerHealthCheck:** **serverReadinessScript**	指定 pod 就绪度健康检查逻辑的字符串 如果没有指定此参数，JWS Operator 会使用 OpenShift 内部 registry 来检查 **http://localhost:8080/health**。 例如： **serverReadinessScript: /bin/bash -c " /usr/bin/curl --noproxy '*' -s 'http://localhost:8080/health' | /usr/bin/grep -i 'status prerequisitesUP"'**
WebImage: **webServerHealthCheck:** **serverLivenessScript**	指定 Pod 存活度健康检查逻辑的字符串 这个参数是可选的。

参数名称	描述
webImageStream	控制 JWS Operator 如何使用提供镜像运行或构建的镜像流的一组参数 JWS Operator 使用镜像流中的最新镜像。 此参数包含 **applicationImage**,**imagePullSecret**,**webApp**, 和 **webServerHealthCheck** 字段。
webImageStream: **imageStreamName**	您创建的镜像流的名称，以允许 JWS Operator 查找基础镜像 例如： **imageStreamName: my-image-name-imagestream:latest**
webImageStream: **imageStreamNamespace**	创建镜像流的命名空间或项目 例如： **imageStreamNamespace: my-namespace**
webImageStream: **webSources**	一组参数，用于描述应用程序源文件所在的位置以及如何构建它们 如果没有指定 **webSources** 参数，JWS Operator 会在镜像流中部署最新的镜像。 此参数包含 **sourceRepositoryUrl**,**sourceRepositoryRef**,**contextDir**, 和 **webSourcesParams** 字段。
webImageStream: **webSources:** **sourceRepositoryUrl**	应用程序源文件所在的 URL 源应包含 Maven **pom.xml** 文件，以支持 Maven 构建。当 Maven 为应用程序生成 **.war** 文件时，**.war** 文件将复制到 JWS Operator 用于部署应用程序的镜像的 **webapps** 目录中（例如 **/opt/jws-5.x/tomcat/webapps**）。 例如： **sourceRepositoryUrl: 'https://github.com/$user/demo-webapp.git'**
webImageStream: **webSources:** **sourceRepositoryRef**	JWS Operator 使用的源存储库的分支 例如： **sourceRepositoryRef: main**

参数名称	描述
webImageStream: webSources: contextDir	运行 **pom.xml** 文件所在的源存储库中的子目录，并且运行 **mvn install** 命令 例如： **contextDir: /**
webImageStream: webSources: webSourcesParams	描述如何构建应用程序镜像的一组参数 这个参数是可选的。 此参数包含 **mavenMirrorUrl**,**artifactDir**,**genericWebHookSecret**, 和 **githubWebHookSecret** 字段。
webImageStream: webSources: webSourcesParams: mavenMirrorUrl	Maven 用于构建 Web 应用的 Maven 代理 URL 如果集群无法访问互联网，则需要此参数。
webImageStream: webSources: webSourcesParams: artifactDir	Maven 存储 Maven 为 Web 应用生成的 **.war** 文件的目录 此目录的内容复制到 JWS Operator 用于部署应用程序的镜像的 **webapps** 目录中（例如：**/opt/jws-5.x/tomcat/webapps**）。 默认值为 **target**。
webImageStream: webSources: webSourcesParams: genericWebHookSecret	触发构建的通用 Webhook 的 secret 名称 有关创建 secret 的更多信息，请参阅为通用或 GitHub Webhook 创建 secret。 有关使用通用 Webhook 的更多信息，请参阅 Webhook 触发器。 例如： **genericWebHookSecret: my-secret**

参数名称	描述
webImageStream: webSources: webSourcesParams: githubWebHookSecret	可以触发构建的 GitHub Webhook 的 secret 名称 有关创建 secret 的更多信息，请参阅为通用或 GitHub Webhook 创建 secret。 有关使用 GitHub Webhook 的更多信息，请参阅 Webhook 触发器。  **注意** 您无法对 GitHub Webhook 执行手动测试。GitHub 生成有效负载，而不为空。
webImageStream: webServerHealthCheck	JWS Operator 使用的健康检查 默认行为是使用健康 valve，它不需要任何参数。 此参数包含 **serverReadinessScript** 和 **serverLivenessScript** 字段。
webImageStream: webServerHealthCheck: serverReadinessScript	指定 pod 就绪度健康检查逻辑的字符串 如果没有指定此参数，JWS Operator 会使用 OpenShift 内部 registry 来检查 **http://localhost:8080/health**。 例如： **serverReadinessScript: /bin/bash -c "/usr/bin/curl --noproxy '*' -s 'http://localhost:8080/health' | /usr/bin/grep -i 'status prerequisitesUP'"**
webImageStream: webServerHealthCheck: serverLivenessScript	指定 Pod 存活度健康检查逻辑的字符串 这个参数是可选的。