Red Hat JBoss Web Server 6.0

Red Hat JBoss Web Server for OpenShift

安装和使用 Red Hat JBoss Web Server for OpenShift

Last Updated: 2024-06-18

Red Hat JBoss Web Server 6.0 Red Hat JBoss Web Server for OpenShift

安装和使用 Red Hat JBoss Web Server for OpenShift

法律通告

摘要

使用 Red Hat JBoss Web Server for OpenShift 指南

目录

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。

第 1 章 RED HAT JBOSS WEB SERVER FOR OPENSHIFT

Red Hat JBoss Web Server (JWS) 6.0 的 Apache Tomcat 10 组件作为专为 Red Hat OpenShift 设计的容器化镜像提供。您可以使用此镜像构建、扩展和测试 Java Web 应用程序，以便在混合云环境中进行部署。

1.1. RED HAT JBOSS WEB SERVER 和 JWS FOR OPENSHIFT 的不同

JWS for OpenShift 镜像与 Red Hat JBoss Web Server 的常规发行版本不同。

对于 OpenShift 的镜像和标准 JBoss Web 服务器部署，请考虑以下区别：

- 在 JWS for OpenShift 镜像中，**/opt/jws-6.0/** 目录是 ***JWS_HOME*** 的位置。
- 在用于 OpenShift 部署的 JWS 中，所有负载均衡都由 OpenShift 路由器而不是 JBoss Core Services **mod_cluster** 连接器或 **mod_jk** 连接器处理。

其他资源

- Red Hat JBoss Web Server 文档

1.2. OPENSHIFT 镜像版本兼容性和支持

OpenShift 镜像通过不同的操作系统版本、配置和接口点进行测试，它们代表 Red Hat OpenShift Container Platform 客户正在使用的技术最常见组合。

其他资源

- OpenShift Container Platform Tested 3.X Integrations 页
- OpenShift Container Platform Tested 4.X Integrations 页

1.3. JBOSS WEB SERVER 支持的构架

JBoss Web Server 支持以下构架：

- AMD64 (x86_64)
- OpenShift 环境中的 IBM Z (s390x)
- OpenShift 环境中的 IBM Power (ppc64le)
- OpenShift 环境中的 ARM64 (aarch64)

您可以在所有支持的构架中使用 OpenJDK 17 的 JBoss Web Server 镜像。有关镜像的更多信息，请参阅 Red Hat Container Catalog 。

其他资源

- 红帽容器目录

1.4. 对红帽容器镜像进行健康检查

所有 OpenShift Container Platform 镜像都关联了一个健康评级。您可以通过进入到 Certfied 容器镜像页面，然后搜索 JBoss Web Server 并选择 6.0 版本来查找 Red Hat **JBoss Web Server** 的健康状况评级。

您还可以对 OpenShift 容器执行健康检查，以测试容器的存活状态和就绪状态。

其他资源

- 使用健康检查来监控应用程序的健康状态

1.5. 其他资源（或后续步骤）

- 支持 Red Hat OpenShift 中的 Red Hat Middleware 产品和组件

第 2 章 RED HAT JBOSS WEB SERVER FOR OPENSHIFT 入门

您可以从 Red Hat 容器 registry 中导入最新的 Red Hat JBoss Web Server for OpenShift 镜像流和模板。之后，您可以使用 OpenShift Source-to-Image (S2I) 流程的 JWS，使用现有的 maven 二进制文件或从源代码为 OpenShift 应用创建 JBoss Web 服务器。

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。

注意

用于 OpenShift 应用程序模板的 JWS 为 Tomcat 10 发布。

2.1. 为 RED HAT CONTAINER REGISTRY 配置身份验证令牌

在导入并使用 Red Hat JBoss Web Server for OpenShift 镜像前，您必须首先确保已配置了身份验证令牌来访问 Red Hat Container Registry。

您可以使用 registry 服务帐户创建身份验证令牌。这意味着您不必在 OpenShift 配置中使用或存储您的红帽帐户用户名和密码。

流程

1. 按照红帽客户门户网站中的说明，使用 registry 服务帐户创建身份验证令牌。
2. 在令牌的 **Token Information** 页面中，点 **OpenShift Secret** 选项卡并下载包含令牌的 OpenShift secret 的 YAML 文件。
3. 使用您下载的 YAML 文件，为 OpenShift 项目创建身份验证令牌机密。
   例如：

```
oc create -f 1234567_myserviceaccount-secret.yaml
```

4. 要为 OpenShift 项目配置 secret，请输入以下命令：

```
oc secrets link default 1234567-myserviceaccount-pull-secret --for=pull
oc secrets link builder 1234567-myserviceaccount-pull-secret --for=pull
```

注意

在前面的示例中，将 **1234567-myserviceaccount** 替换为您在上一步中创建的 secret 的名称。

其他资源

- Red Hat Container Registry Authentication 网页
- 允许 Pod 引用其他安全 registry 中的镜像

2.2. 导入 JBOSS WEB 服务器镜像流和模板

您可以从 Red Hat Container Registry 中导入 Red Hat JBoss Web Server for OpenShift 镜像流和模板。您必须将 JDK 的最新 JBoss Web 服务器镜像流和模板导入到 OpenShift 项目的命名空间。

先决条件

- 您已为 Red Hat Container Registry 配置了身份验证令牌 。

流程

1. 使用您的客户门户网站凭证登录到 Red Hat Container Registry。如需更多信息，请参阅 Red Hat Container Registry 身份验证。
2. 要为 OpenJDK 17 导入镜像流，请输入以下命令：

```
for resource in \
jws60-openjdk17-tomcat10-ubi8-basic-s2i.json \
jws60-openjdk17-tomcat10-ubi8-https-s2i.json \
jws60-openjdk17-tomcat10-ubi8-image-stream.json
do
oc replace -n openshift --force -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-webserver-6-openshift-image/jws60el8-v6.0.0/templates/${resource}
done
```

前面的命令导入 UBI8 JDK 17 镜像流、**jboss-webserver60-openjdk17-tomcat10-openshift-ubi8** 以及命令中指定的所有模板。

2.3. 为 OPENSHIFT 镜像导入最新的 JWS

您可以使用 **import-image** 命令导入 OpenShift 镜像的最新可用 JWS。红帽为 OpenJDK 17 提供了一个 JWS for OpenShift 镜像，带有 JBoss Web Server 6.0 发行版本。

先决条件

- 登录到 Red Hat Container Registry 。
- 您已导入了镜像流和模板。

流程

- 要使用 OpenJDK 17 OpenShift 镜像更新核心 JBoss Web Server 6.0 tomcat 10，请输入以下命令：

```
$ oc -n openshift import-image \
  jboss-webserver60-openjdk17-tomcat10-openshift-ubi8:6.0.0
```

注意

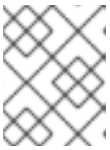
您导入的每个镜像末尾的 **6.0.0** 标签指的是镜像流中设置的 流 版本。

2.4. 用于 OPENSHIFT S2I 流程的 JWS

您可以使用带有应用程序模板参数和环境变量的 OpenShift Source-to-image (S2I) 进程，为 OpenShift 镜像运行和配置 JWS。

用于 OpenShift 镜像的 JWS 的 S2I 流程可以正常工作：

- 如果 **configuration** 源目录包含 Maven **settings.xml** 文件，则 **settings.xml** 文件将移到新镜像的 ***$HOME*/.m2/** 目录中。
- 如果源存储库包含 **pom.xml** 文件，则使用 **$MAVEN_ARGS** 环境变量的内容触发 Maven 构建。默认情况下，**package** 目标与 **openshift** 配置集一同使用，其中包括用于跳过测试的 **-DskipTests** 参数，以及用于启用 Red Hat GA 存储库的 **-Dcom.redhat.xpaas.repo.redhatga** 参数。
- 成功 Maven 构建的结果被复制到 **/opt/jws-6.0/tomcat/webapps** 目录中。这包括由 **$ARTIFACT_DIR** 环境变量指定的源目录中的所有 WAR 文件。**$ARTIFACT_DIR** 的默认值为 **target/** 目录。
  您可以使用 **$MAVEN_ARGS_APPEND** 环境变量修改 Maven 参数。
- **deployments** 源目录中的所有 WAR 文件都复制到 **/opt/jws-6.0/tomcat/webapps** 目录中。
- **配置** 源目录中的所有文件都复制到 **/opt/jws-6.0/tomcat/conf/** 目录中，不包括 Maven **settings.xml** 文件。
- **lib** 源目录中的所有文件都复制到 **/opt/jws-6.0/tomcat/lib/** 目录中。



注意

如果要使用自定义 Tomcat 配置文件，请使用与常规 Tomcat 安装相同的文件名，如 **context.xml** 和 **server.xml**。

有关配置 S2I 流程以使用自定义 Maven 工件存储库镜像的更多信息，请参阅 Maven 工件存储库镜像以及 OpenShift 的 JWS。

其他资源

- Apache Maven 项目网站

2.5. 使用现有 MAVEN 二进制文件为 OPENSHIFT 应用创建 JWS

您可以使用现有的 Maven 二进制文件为 OpenShift 应用创建 JWS。您可以使用 **oc start-build** 命令在 OpenShift 上部署现有应用程序。

注意

此流程演示了如何创建基于 tomcat-websocket-chat quickstart 示例的示例应用程序。

先决条件

- 已有您要部署在 JWS for OpenShift 中的应用程序的 **.war**, **.ear**, 或 **.jar** 文件，或已在本地构建了应用程序。
  例如，要在本地构建 **tomcat-websocket-chat** 应用程序，请执行以下步骤：

1. 要克隆源代码，请输入以下命令：

```
$ git clone https://github.com/jboss-openshift/openshift-quickstarts.git
```

2. 配置Red Hat JBoss Middleware Maven 存储库，参阅 配置 Red Hat JBoss Middleware Maven 存储库。

有关 Maven 存储库的更多信息，请参阅 Red Hat JBoss Ener Enterprise Maven Repository 网页。

3. 要构建应用程序，请输入以下命令：

```
$ cd openshift-quickstarts/tomcat-websocket-chat/
$ mvn clean package
```

前面的命令会产生以下输出：

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Tomcat websocket example 1.2.0.Final
[INFO] ------------------------------------------------------------------------
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:28 min
[INFO] Finished at: 2018-01-16T15:59:16+10:00
[INFO] Final Memory: 19M/271M
[INFO] ------------------------------------------------------------------------
```

流程

1. 在本地文件系统中，为二进制构建创建一个源目录和一个 **deployments** 子目录。
例如，要为 **tomcat-websocket-chat** 应用程序创建 **/ocp** 源目录和 **/deployments** 子目录，请输入以下命令：

```
$ cd openshift-quickstarts/tomcat-websocket-chat/
$ mkdir -p ocp/deployments
```

注意

源目录可以包含未包含在 Maven 二进制文件中的应用程序所需的任何内容。如需更多信息，请参阅 JWS for OpenShift S2I 流程。

2. 将 **.war**、**.ear** 或 **.jar** 二进制文件复制到 **deployments** 子目录。
例如，要复制 tomcat-websocket-chat 应用程序的 **.war** 文件，请输入以下命令：

```
$ cp target/websocket-chat.war ocp/deployments/
```

注意

在上例中，**target/websocket-chat.war** 是您要复制的二进制文件的路径。

源目录的 **deployments** 子目录中的应用存档复制到 OpenShift 上构建的镜像的 **$JWS_HOME/tomcat/webapps/** 目录中。要成功部署应用程序，您必须确保包含 Web 应用数据的目录层次结构是正确的。如需更多信息，请参阅 JWS for OpenShift S2I 流程。

3. 登录到 OpenShift 实例：

```
$ oc login <url>
```

4. 如果需要，创建一个新项目。
例如：

```
$ oc new-project jws-bin-demo
```

注意

在前面的示例中，**jws-bin-demo** 是您要创建的项目名称。

5. 识别应用程序要使用的 JWS for OpenShift 镜像流：

```
$ oc get is -n openshift | grep ^jboss-webserver | cut -f1 -d ' '
```

前面的命令会产生以下类型的输出：

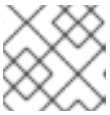
```
jboss-webserver60-openjdk17-tomcat10-openshift-ubi8
```

注意

-n openshift 选项指定要使用的项目。**oc get is -n openshift** 命令从 **openshift** 项目中获取镜像流资源。

6. 创建新构建配置，并确保您指定镜像流和应用程序名称。
例如，要为 tomcat-websocket-chat 应用程序创建新构建配置，请执行以下操作：

```
$ oc new-build --binary=true \
 --image-stream=jboss-webserver60-openjdk17-tomcat10-openshift-ubi8:latest\
 --name=jws-wsch-app
```



注意

在前面的示例中，**jws-wsch-app** 是 JWS for OpenShift 应用的 JWS 的名称。

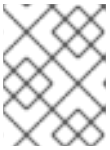
前面的命令会产生以下类型的输出：

```
--> Found image 8c3b85b (4 weeks old) in image stream "openshift/jboss-webserver60-
tomcat10-openshift" under tag "latest" for "jboss-webserver60"

    JBoss Web Server 6.0
    --------------------
    Platform for building and running web applications on JBoss Web Server 6.0 - Tomcat v10

    Tags: builder, java, tomcat10

    * A source build using binary input will be created
      * The resulting image will be pushed to image stream "jws-wsch-app:latest"
      * A binary build was created, use 'start-build --from-dir' to trigger a new build
```

```
--> Creating resources with label build=jws-wsch-app ...
    imagestream "jws-wsch-app" created
    buildconfig "jws-wsch-app" created
--> Success
```

7. 启动二进制构建。
例如：

```
$ oc start-build jws-wsch-app --from-dir=./ocp --follow
```



注意

在前面的示例中，**jws-wsch-app** 是 JWS for OpenShift 应用的名称， **ocp** 是源目录的名称。

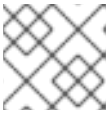
前面的命令指示 OpenShift 使用您为 OpenShift 镜像构建的二进制输入创建的源目录。

前面的命令会产生以下类型的输出：

```
Uploading directory "ocp" as binary input for the build ...
build "jws-wsch-app-1" started
Receiving source from STDIN as archive ...

Copying all deployments war artifacts from /home/jboss/source/deployments directory into
`/opt/jws-6.0/tomcat/webapps` for later deployment...
'/home/jboss/source/deployments/websocket-chat.war' -> '/opt/jws-
6.0/tomcat/webapps/websocket-chat.war'

Pushing image 172.30.202.111:5000/jws-bin-demo/jws-wsch-app:latest ...
Pushed 0/7 layers, 7% complete
Pushed 1/7 layers, 14% complete
Pushed 2/7 layers, 29% complete
Pushed 3/7 layers, 49% complete
Pushed 4/7 layers, 62% complete
Pushed 5/7 layers, 92% complete
Pushed 6/7 layers, 100% complete
Pushed 7/7 layers, 100% complete
Push successful
```

8. 根据镜像创建新 OpenShift 应用程序：
例如：

```
$ oc new-app jws-wsch-app
```



注意

在前面的示例中，**jws-wsch-app** 是 JWS for OpenShift 应用的 JWS 的名称。

前面的命令会产生以下类型的输出：

```
--> Found image e5f3a6b (About a minute old) in image stream "jws-bin-demo/jws-wsch-app"
```

```
under tag "latest" for "jws-wsch-app"

    JBoss Web Server 6.0
    --------------------
    Platform for building and running web applications on JBoss Web Server 6.0 - Tomcat v10

    Tags: builder, java, tomcat10

    * This image will be deployed in deployment config "jws-wsch-app"
    * Ports 8080/tcp, 8443/tcp, 8778/tcp will be load balanced by service "jws-wsch-app"
      * Other containers can access this service through the hostname "jws-wsch-app"

--> Creating resources ...
    deploymentconfig "jws-wsch-app" created
    service "jws-wsch-app" created
--> Success
    Application is not exposed. You can expose services to the outside world by executing one
or more of the commands below:
     'oc expose svc/jws-wsch-app'
    Run 'oc status' to view your app.
```

9. 公开该服务以使应用程序可以被用户访问：
例如，要使 **jws-wsch-app** 应用程序可以被访问，请执行以下步骤：

   a. 检查要公开的服务名称：

   ```
   $ oc get svc -o name
   ```

   前面的命令会产生以下类型的输出：

   ```
   service/jws-wsch-app
   ```

   b. 公开服务：

   ```
   $ oc expose svc/jws-wsch-app
   ```

   前面的命令会产生以下类型的输出：

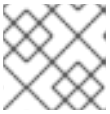
   ```
   route "jws-wsch-app" exposed
   ```

10. 检索公开路由的地址：

```
oc get routes --no-headers -o custom-columns='host:spec.host' jws-wsch-app
```

11. 打开 Web 浏览器，再输入 URL 以访问应用。
例如，要访问示例 **jws-wsch-app** 应用程序，请输入以下 URL：

\http://*<address_of_exposed_route>*/websocket-chat



注意

在前面的示例中，将 ***<address_of_exposed_route>*** 替换为您的部署的适当值。

其他资源

- **oc start-build** 命令

2.6. 从源代码创建一个 JWS FOR OPENSHIFT 应用程序

您可以通过源代码创建一个 JWS for OpenShift 应用程序。

有关通过源代码创建新的 OpenShift 应用程序的详细信息，请参阅 OpenShift.com - 从源代码创建应用程序。

先决条件

- 应用程序数据结构正确。如需更多信息，请参阅 JWS for OpenShift S2I 流程。

流程

1. 登录到 OpenShift 实例：

```
$ oc login <url>
```

2. 根据需要创建新项目：

```
$ oc new-project <project-name>
```

注意

在前面的示例中，将 ***<project-name>*** 替换为您要创建的项目名称。

3. 识别应用程序要使用的 JWS for OpenShift 镜像流：

```
$ oc get is -n openshift | grep ^jboss-webserver | cut -f1 -d ' '
```

前面的命令会产生以下类型的输出：

```
jboss-webserver60-openjdk17-tomcat10-openshift-ubi8
```

注意

-n openshift 选项指定要使用的项目。**oc get is -n openshift** 命令从 **openshift** 项目中获取镜像流资源。

4. 使用 Red Hat JBoss Web Server for OpenShift 镜像从源代码创建新 OpenShift 应用程序：

```
$ oc new-app \
 <source_code_location>\
 --image-stream=jboss-webserver60-openjdk17-tomcat10-openshift-ubi8\
 --name=<openshift_application_name>
```

例如：

```
$ oc new-app \
 \https://github.com/jboss-openshift/openshift-quickstarts.git#main \
 --image-stream=jboss-webserver60-openjdk17-tomcat10-openshift-ubi8\
 --context-dir='tomcat-websocket-chat' \
 --name=jws-wsch-app
```

前面的命令将源代码添加到镜像中，并编译源代码。前面的命令还会创建构建配置和服务。

5. 要公开应用程序，请执行以下步骤：

   a. 检查要公开的服务名称：

   ```
   $ oc get svc -o name
   ```

   前面的命令会产生以下类型的输出：

   ```
   service/<openshift_application_name>
   ```

   b. 公开服务：

   ```
   $ oc expose svc/<openshift_application_name>
   ```

   前面的命令会产生以下类型的输出：

   ```
   route "<openshift_application_name>" exposed
   ```

6. 检索公开路由的地址：

   ```
   oc get routes --no-headers -o custom-columns='host:spec.host'
   <openshift_application_name>
   ```

7. 打开 Web 浏览器，并输入以下 URL 来访问应用程序：
   \http://*<address_of_exposed_route>*/*<java_application_name>*

注意

在前面的示例中，将 ***<address_of_exposed_route>*** 和 ***<java_application_name>*** 替换为部署的相应值。

2.7. 在 TOMCAT/LIB 目录中添加额外的 JAR 文件

您可以使用 Docker 在 **tomcat/lib** 目录中添加其他 Java 存档 (JAR) 文件。

流程

1. 启动 Docker 中的镜像：

   ```
   docker run --network host -i -t -p 8080:8080 ImageURL
   ```

2. 查找 **CONTAINER ID** ：

   ```
   docker ps | grep <ImageName>
   ```

3. 将库复制到 **tomcat/lib/** 目录中：

```
docker cp <yourLibrary> <CONTAINER ID>:/opt/jws-6.0/tomcat/lib/
```

4. 将更改提交到新镜像：

```
docker commit <CONTAINER ID> <NEW IMAGE NAME>
```

5. 创建新镜像标签：

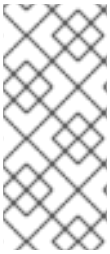
```
docker tag <NEW IMAGE NAME>:latest <NEW IMAGE REGISTRY URL>:<TAG>
```

6. 将镜像推送到 registry：

```
docker push <NEW IMAGE REGISTRY URL>
```

第 3 章 RED HAT OPENSHIFT 的 RED HAT JBOSS WEB SERVER METERING 标签

您可以在 Red Hat JBoss Web Server pod 中添加 metering 标签，并使用 OpenShift Metering Operator 检查红帽订阅详情。



注意

- 不要将 metering 标签添加到 Operator 或模板部署和管理的任何 pod 中。
- 您可以使用 OpenShift Container Platform 版本 4.8 及更早版本上的 Metering Operator 将标签应用到 pod。从 4.9 版本中，在没有直接替换的情况下，Metering Operator 不再可用。

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**

其他资源

- 在 OpenShift Container Platform 中配置和使用 Metering

附录 A. S2I 脚本和 MAVEN

Red Hat JBoss Web Server for OpenShift 镜像包括 S2I 脚本和 Maven。

A.1. MAVEN 工件存储库镜像以及 JWS FOR OPENSHIFT

Maven 存储库包含构建工件和依赖项，如项目 Java 存档(JAR)文件、库 JAR 文件、插件或其他特定于项目的工件。Maven 存储库还可定义在执行 source-to-image (S2I) 构建时可从中下载工件的位置。除了使用 Maven Central 存储库 外，一些机构还会部署本地自定义存储库(mirror)。

本地镜像提供以下优点：

- 通过同步的镜像，使地理位置更近，速度更快
- 对存储库内容进行更大的控制
- 能够在不同团队（开发人员和持续集成(CI)）共享工件，而无需依赖公共服务器和存储库
- 改进了构建时间

Maven 存储库管理器可以充当本地缓存到镜像。如果已部署了存储库管理器，且可以在指定的 URL 位置访问外部，则 S2I 构建可以使用此存储库。您可以通过在应用的构建配置中添加 **MAVEN_MIRROR_URL** 环境变量来使用内部 Maven 存储库。

其他资源

- Apache Maven 项目：最佳实践 - 使用存储库管理器

A.1.1. 使用内部 Maven 存储库进行新构建配置

您可以在 **oc new-app** 命令或 **oc new-build** 命令中指定 **--build-env** 选项，将 **MAVEN_MIRROR_URL** 环境变量添加到应用程序的新构建配置中。

流程

1. 输入以下命令：

```
$ oc new-app \
 https://github.com/jboss-openshift/openshift-quickstarts.git#main \
 --image-stream=jboss-webserver60-openjdk17-tomcat10-openshift-ubi8:latest\
 --context-dir='tomcat-websocket-chat' \
 --build-env MAVEN_MIRROR_URL=\http://10.0.0.1:8080/repository/internal/ \
 --name=jws-wsch-app
```

注意

前面的命令假定已部署了存储库管理器，可通过 **http://10.0.0.1:8080/repository/internal/** 访问。

A.1.2. 将内部 Maven 存储库用于现有构建配置

您可以使用 **oc env** 命令指定构建配置的名称，将 **MAVEN_MIRROR_URL** 环境变量添加到应用程序的现有构建配置中。

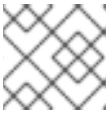
流程

1. 识别需要 **MAVEN_MIRROR_URL** 变量的构建配置：

```
$ oc get bc -o name
```

前面的命令会产生以下类型的输出：

```
buildconfig/jws
```



注意

在前面的示例中，jws 是构建配置的名称。

2. 将 **MAVEN_MIRROR_URL** 环境变量添加到 **buildconfig/jws** ：

```
$ oc env bc/jws MAVEN_MIRROR_URL="http://10.0.0.1:8080/repository/internal/"

buildconfig "jws" updated
```

3. 验证构建配置是否已更新：

```
$ oc env bc/jws --list

# buildconfigs jws
MAVEN_MIRROR_URL=http://10.0.0.1:8080/repository/internal/
```

4. 使用 **oc start-build** 调度应用程序的新构建

注意

在应用程序构建过程中，从存储库管理器下载 Maven 依赖项，而不是从默认公共存储库下载。构建过程完成后，镜像包含构建过程中检索和使用的所有依赖项。

A.2. RED HAT JBOSS WEB SERVER FOR OPENSHIFT 镜像中包含的脚本

Red Hat JBoss Web Server for OpenShift 镜像包括运行 Catalina 的脚本，并使用 Maven 创建和部署 **.war** 软件包。

run

运行 Catalina (Tomcat)

assemble

使用 Maven 构建 Web 应用源，创建 **.war** 文件，并将 **.war** 文件移动到 **$JWS_HOME/tomcat/webapps** 目录。

A.3. JWS FOR OPENSHIFT 数据源

JWS for OpenShift 提供了三种数据源类型：

默认内部数据源

默认情况下，PostgreSQL、MySQL 和 MongoDB 数据源可以通过 Red Hat Registry 在 OpenShift 上获得。这些数据源不需要为镜像流配置额外的环境文件。要启用数据库被发现和用作数据源，您可以将 **DB_SERVICE_PREFIX_MAPPING** 环境变量设置为 OpenShift 服务的名称。

其他内部数据源

这些数据源在 OpenShift 上运行，但默认情况下无法通过 Red Hat Registry 提供它们。添加到 OpenShift Secret 的环境文件提供了其他内部数据源的配置。

外部数据源

这些数据源不在 OpenShift 中运行。添加到 OpenShift Secret 的环境文件可以提供外部数据源的配置。

ENV_FILES 属性

您可以将数据源的环境变量添加到项目的 OpenShift Secret 中。您可以使用 **ENV_FILES** 属性在模板中调用这些环境文件。

DB_SERVICE_PREFIX_MAPPING 环境变量

数据源根据特定环境变量的值自动创建。**DB_SERVICE_PREFIX_MAPPING** 环境变量定义数据源的 JNDI 映射。

DB_SERVICE_PREFIX_MAPPING 变量允许的值是以逗号分隔的 **POOLNAME-DATABASETYPE=PREFIX** triplets 列表。每个 triplet 都包含以下值：

- **POOLNAME** 用作数据源中的 **pool-name**。
- **DATABASETYPE** 是要使用的数据库驱动程序。
- **PREFIX** 是用来配置数据源的环境变量名称中的前缀。

对于每个在 **DB_SERVICE_PREFIX_MAPPING** 环境变量中定义的 **POOLNAME-DATABASETYPE=PREFIX** triplet，启动脚本都会创建一个单独的数据源，在运行镜像时执行。

其他资源

- 数据源配置环境变量

A.4. 用于 OPENSHIFT 兼容环境变量的 JWS

您可以使用 source-to-image (S2I) **build** 命令包含环境变量来修改构建配置。如需更多信息，请参阅 Maven 工件存储库镜像和 JWS for OpenShift。

下表列出了 Red Hat JBoss Web Server for OpenShift 镜像的有效环境变量：

变量名称	显示名称	描述	值示例
ARTIFACT_DIR	不适用	此目录中的 **.war**、**.ear**、和 **.jar** 文件将复制到 **deployments** 目录中	target
APPLICATION_NAME	应用程序名称	应用程序的名称	jws-app

变量名称	显示名称	描述	值示例
CONTEXT_DIR	上下文目录	构建 Git 项目中的路径；根项目目录为空	tomcat-websocket-chat
GITHUB_WEBHOOK_SECRET	GitHub Webhook Secret	GitHub 触发器 secret	表达式包括：[a-zA-Z0-9]{8}
GENERIC_WEBHOOK_SECRET	通用 Webhook Secret	通用构建触发器 secret	表达式包括：[a-zA-Z0-9]{8}
HOSTNAME_HTTP	自定义 HTTP 路由主机名	http 服务路由的自定义主机名。为默认主机名留空	<application-name>-<project>.<default-domain-suffix>
HOSTNAME_HTTPS	自定义 HTTPS 路由主机名	https 服务路由的自定义主机名。为默认主机名留空	<application-name>-<project>.<default-domain-suffix>
IMAGE_STREAM_NAMESPACE	镜像流命名空间	安装 Red Hat Middleware 镜像的 ImageStreams 的命名空间	openshift
JWS_HTTPS_CERTIFICATE	证书文件名	证书文件的名称	rsa-cert.pem
JWS_HTTPS_CERTIFICATE_CHAIN	证书链文件名	证书链文件的名称	ca-chain.cert.pem
JWS_HTTPS_CERTIFICATE_DIR	证书目录名称	存储证书的目录的名称	cert
JWS_HTTPS_CERTIFICATE_KEY	证书密钥文件名	证书密钥文件的名称	rsa-key.pem
JWS_HTTPS_CERTIFICATE_PASSWORD	证书密码	证书密码	P5ssw0rd
SOURCE_REPOSITORY_URL	Git 存储库 URL	应用程序的 Git 源 URI	*https://github.com/jboss-openshift/openshift-quickstarts.git*
SOURCE_REPOSITORY_REFERENCE	Git 参考	Git 分支/标签参考	1.2

变量名称	显示名称	描述	值示例
IMAGE_STREAM_NAMESPACE	镜像流命名空间	安装 Red Hat Middleware 镜像的 ImageStreams 的命名空间	openshift
MAVEN_MIRROR_URL	Maven Mirror URL	用于配置的 Maven 镜像/存储库管理器的 URL。	http://10.0.0.1:8080/repository/internal/

附录 B. 用于 OPENSHIFT 的 JWS 上的 VALVES

您可以定义以下环境变量，将 valve 组件插入到关联的 Catalina 容器的请求处理管道中。

变量名称	描述	值示例	默认值
ENABLE_ACCESS_LOG	启用 Access Log Valve 将访问信息记录到标准输出频道。	*true*	*false*

附录 C. 检查 OPENSHIFT 日志

您可以使用 **oc logs** 命令查看 OpenShift 日志或控制台为正在运行的容器提供的日志。

流程

- 输入以下命令：

```
$ oc logs -f <pod_name> <container_name>
```

注意

在前面的命令中，将 ***<pod_name>*** 和 ***<container_name>*** 替换为您的部署的适当值。

访问日志保存在 **/opt/jws-6.0/tomcat/logs/** 目录中。