Red Hat JBoss Web Server 6.0

Red Hat JBoss Web Server for OpenShift

Installing and using Red Hat JBoss Web Server for OpenShift

Last Updated: 2024-03-15

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

Installing and using Red Hat JBoss Web Server for OpenShift

Legal Notice

Abstract

Guide to using Red Hat JBoss Web Server for OpenShift

Table of Contents

PROVIDING FEEDBACK ON RED HAT JBOSS WEB SERVER DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to **create a ticket**.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. RED HAT JBOSS WEB SERVER FOR OPENSHIFT

The Apache Tomcat 10 component of Red Hat JBoss Web Server (JWS) 6.0 is available as a containerized image that is designed for Red Hat OpenShift. You can use this image to build, scale, and test Java web applications for deployment across hybrid cloud environments.

1.1. DIFFERENCES BETWEEN RED HAT JBOSS WEB SERVER AND JWS FOR OPENSHIFT

JWS for OpenShift images are different from a regular release of Red Hat JBoss Web Server.

Consider the following differences between the JWS for OpenShift images and a standard JBoss Web Server deployment:

- In a JWS for OpenShift image, the **/opt/jws-6.0/** directory is the location of ***JWS_HOME***.
- In a JWS for OpenShift deployment, all load balancing is handled by the OpenShift router rather than the JBoss Core Services **mod_cluster** connector or **mod_jk** connector.

Additional resources

- Red Hat JBoss Web Server documentation

1.2. OPENSHIFT IMAGE VERSION COMPATIBILITY AND SUPPORT

OpenShift images are tested with different operating system versions, configurations and interface points that represent the most common combination of technologies that Red Hat OpenShift Container Platform customers are using.

Additional resources

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

1.3. SUPPORTED ARCHITECTURES FOR JBOSS WEB SERVER

JBoss Web Server supports the following architectures:

- AMD64 (x86_64)
- IBM Z (s390x) in the OpenShift environment
- IBM Power (ppc64le) in the OpenShift environment
- ARM64 (aarch64) in the OpenShift environment

You can use the JBoss Web Server image for OpenJDK 17 with all supported architectures. For more information about images, see the Red Hat Container Catalog .

Additional resources

- Red Hat Container Catalog

1.4. HEALTH CHECKS FOR RED HAT CONTAINER IMAGES

All OpenShift Container Platform images have a health rating associated with them. You can find the health rating for Red Hat JBoss Web Server by navigating to the Certfied container images page, and then search for **JBoss Web Server** and select the 6.0 version.

You can also perform health checks on an OpenShift container to test the container for liveliness and readiness.

Additional resources

- Monitoring application health by using health checks

1.5. ADDITIONAL RESOURCES (OR NEXT STEPS)

- Support of Red Hat Middleware products and components on Red Hat OpenShift

CHAPTER 2. GETTING STARTED WITH RED HAT JBOSS WEB SERVER FOR OPENSHIFT

You can import the latest Red Hat JBoss Web Server for OpenShift image streams and templates from the Red Hat container registry. You can subsequently use the JWS for OpenShift Source-to-Image (S2I) process to create JBoss Web Server for OpenShift applications by using existing maven binaries or from source code.

Before you follow the instructions in this document, you must ensure that an OpenShift cluster is already installed and configured as a prerequisite. For more information about installing and configuring OpenShift clusters, see the OpenShift Container Platform Installing guide.

NOTE

The JWS for OpenShift application templates are distributed for Tomcat 10.

2.1. CONFIGURING AN AUTHENTICATION TOKEN FOR THE RED HAT CONTAINER REGISTRY

Before you can import and use a Red Hat JBoss Web Server for OpenShift image, you must first ensure that you have configured an authentication token to access the Red Hat Container Registry.

You can create an authentication token by using a registry service account. This means that you do not have to use or store your Red Hat account username and password in your OpenShift configuration.

Procedure

1. Follow the instructions on the Red Hat Customer Portal to create an authentication token using a registry service account.
2. On the **Token Information** page for your token, click the **OpenShift Secret** tab and download the YAML file that contains the OpenShift secret for the token.
3. Use the YAML file that you have downloaded to create the authentication token secret for your OpenShift project.
   For example:

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

4. To configure the secret for your OpenShift project, enter the following commands:

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

NOTE

In the preceding examples, replace **1234567-myserviceaccount** with the name of the secret that you created in the previous step.

Additional resources

- Red Hat Container Registry Authentication web page

- Allowing pods to reference images from other secured registries

2.2. IMPORTING JBOSS WEB SERVER IMAGE STREAMS AND TEMPLATES

You can import Red Hat JBoss Web Server for OpenShift image streams and templates from the Red Hat Container Registry. You must import the latest JBoss Web Server image streams and templates for your JDK into the namespace of your OpenShift project.

Prerequisites

- You have configured an authentication token for the Red Hat Container Registry .

Procedure

1. Log in to the Red Hat Container Registry by using your Customer Portal credentials. For more information, see Red Hat Container Registry Authentication .
2. To import the image stream for OpenJDK 17, enter the following command:

```
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
```

The preceding command imports the UBI8 JDK 17 image stream, **jboss-webserver60-openjdk17-tomcat10-openshift-ubi8**, and all templates specified in the command.

2.3. IMPORTING THE LATEST JWS FOR OPENSHIFT IMAGE

You can import the latest available JWS for OpenShift image by using the **import-image** command. Red Hat provides a JWS for OpenShift image for OpenJDK 17 with the JBoss Web Server 6.0 release.

Prerequisites

- You are logged in to the Red Hat Container Registry .
- You have imported image streams and templates.

Procedure

- To update the core JBoss Web Server 6.0 tomcat 10 with OpenJDK 17 OpenShift image, enter the following command:

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

NOTE

The **6.0.0** tag at the end of each image you import refers to the stream version that is set in the image stream.

2.4. JWS FOR OPENSHIFT S2I PROCESS

You can run and configure the JWS for OpenShift images by using the OpenShift source-to-image (S2I) process with the application template parameters and environment variables.

The S2I process for the JWS for OpenShift images works as follows:

- If the **configuration** source directory contains a Maven **settings.xml** file, the **settings.xml** file is moved to the ***$HOME*/.m2/** directory of the new image.
- If the source repository contains a **pom.xml** file, a Maven build is triggered using the contents of the **$MAVEN_ARGS** environment variable.
  By default, the **package** goal is used with the **openshift** profile, which includes the **-DskipTests** argument to skip tests, and the **-Dcom.redhat.xpaas.repo.redhatga** argument to enable the Red Hat GA repository.
- The results of a successful Maven build are copied to the **/opt/jws-6.0/tomcat/webapps** directory. This includes all WAR files from the source directory that is specified by the **$ARTIFACT_DIR** environment variable. The default value of **$ARTIFACT_DIR** is the **target/** directory.
  You can use the **$MAVEN_ARGS_APPEND** environment variable to modify the Maven arguments.
- All WAR files from the **deployments** source directory are copied to the **/opt/jws-6.0/tomcat/webapps** directory.
- All files in the **configuration** source directory are copied to the **/opt/jws-6.0/tomcat/conf/** directory, excluding the Maven **settings.xml** file.
- All files in the **lib** source directory are copied to the **/opt/jws-6.0/tomcat/lib/** directory.

NOTE

If you want to use custom Tomcat configuration files, use the same file names that are used for a normal Tomcat installation such as **context.xml** and **server.xml**.

For more information about configuring the S2I process to use a custom Maven artifacts repository mirror, see Maven artifact repository mirrors and JWS for OpenShift .

Additional resources

- Apache Maven Project website

2.5. CREATING A JWS FOR OPENSHIFT APPLICATION BY USING EXISTING MAVEN BINARIES

You can create a JWS for OpenShift application by using existing Maven binaries. You can use the **oc start-build** command to deploy existing applications on OpenShift.

NOTE

This procedure shows how to create an example application that is based on the tomcat-websocket-chat quickstart example.

Prerequisites

- You have an existing **.war**, **.ear**, or **.jar** file for the application that you want to deploy on JWS for OpenShift or you have built the application locally.
  For example, to build the **tomcat-websocket-chat** application locally, perform the following steps:

  1. To clone the source code, enter the following command:

     ```
     $ git clone https://github.com/web-servers/tomcat-websocket-chat-quickstart.git
     ```

  2. Configure the Red Hat JBoss Middleware Maven repository, as described in Configure the Red Hat JBoss Middleware Maven Repository.
     For more information about the Maven repository, see the Red Hat JBoss Enerprise Maven Repository web page.

  3. To build the application, enter the following commands:

     ```
     $ cd tomcat-websocket-chat-quickstart/tomcat-websocket-chat/
     $ mvn clean package
     ```

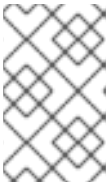
     The preceding command produces the following output:

     ```
     [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] ------------------------------------------------------------------------
     ```

Procedure

1. On your local file system, create a source directory for the binary build and a **deployments** subdirectory.
   For example, to create a **/ocp** source directory and a **/deployments** subdirectory for the **tomcat-websocket-chat** application, enter the following commands:

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



NOTE

The source directory can contain any content required by your application that is not included in the Maven binary. For more information, see JWS for OpenShift S2I process.

2. Copy the **.war**,**.ear**, or **.jar** binary files to the **deployments** subdirectory.
For example, to copy the **.war** file for the example tomcat-websocket-chat application, enter the following command:

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

NOTE

In the preceding example, **target/websocket-chat.war** is the path to the binary file you want to copy.

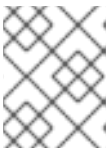
Application archives in the **deployments** subdirectory of the source directory are copied to the **$JWS_HOME/tomcat/webapps/** directory of the image that is being built on OpenShift. To allow the application to be deployed successfully, you must ensure that the directory hierarchy that contains the web application data is structured correctly. For more information, see JWS for OpenShift S2I process.

3. Log in to the OpenShift instance:

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

4. Create a new project if required.
For example:

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



NOTE

In the preceding example, **jws-bin-demo** is the name of the project you want to create.

5. Identify the JWS for OpenShift image stream to use for your application:

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

The preceding command produces the following type of output:

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

NOTE

The **-n openshift** option specifies the project to use. The **oc get is -n openshift** command gets the image stream resources from the **openshift** project.

6. Create the new build configuration, and ensure that you specify the image stream and application name.
   For example, to create the new build configuration for the example tomcat-websocket-chat application:

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

NOTE

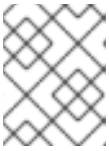
In the preceding example, **jws-wsch-app** is the name of the JWS for OpenShift application.

The preceding command produces the following type of output:

```
--> 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. Start the binary build.
   For example:

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



NOTE

In the preceding example, **jws-wsch-app** is the name of the JWS for OpenShift application, and **ocp** is the name of the source directory.

The preceding command instructs OpenShift to use the source directory that you have created for binary input of the OpenShift image build.

The preceding command produces the following type of output:

```
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. Create a new OpenShift application based on the image:
   For example:

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

NOTE

In the preceding example, **jws-wsch-app** is the name of the JWS for OpenShift application.

The preceding command produces the following type of output:

```
--> 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. Expose the service to make the application accessible to users:
   For example, to make the example **jws-wsch-app** application accessible, perform the following steps:

a. Check the name of the service to expose:

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

The preceding command produces the following type of output:

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

b. Expose the service:

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

The preceding command produces the following type of output:

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

10. Retrieve the address of the exposed route:

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

11. Open a web browser and enter the URL to access the application.
For example, to access the example **jws-wsch-app** application, enter the following URL:

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

NOTE

In the preceding example, replace ***<address_of_exposed_route>*** with the appropriate value for your deployment.

Additional resources

- **oc start-build** command

2.6. CREATING A JWS FOR OPENSHIFT APPLICATION FROM SOURCE CODE

You can create a JWS for OpenShift application from source code.

For detailed information about creating new OpenShift applications from source code, see OpenShift.com - Creating an application from source code .

Prerequisites

- The application data is structured correctly. For more information, see JWS for OpenShift S2I process.

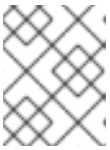
Procedure

1. Log in to the OpenShift instance:

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

2. Create a new project if required:

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



NOTE

In the preceding example, replace ***<project-name>*** with the name of the project you want to create.

3. Identify the JWS for OpenShift image stream to use for your application:

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

The preceding command produces the following type of output:

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

NOTE

The **-n openshift** option specifies the project to use. The **oc get is -n openshift** command gets the image stream resources from the **openshift** project.

4. Create the new OpenShift application from source code by using Red Hat JBoss Web Server for OpenShift images:

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

For example:

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

The preceding command adds the source code to the image and compiles the source code. The preceding command also creates the build configuration and services.

5. To expose the application, perform the following steps:

   a. To check the name of the service to expose:

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

   The preceding command produces the following type of output:

   ```
   service/<openshift_application_name>
   ```

   b. To expose the service:

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

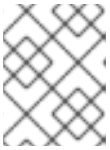
The preceding command produces the following type of output:

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

6. To retrieve the address of the exposed route:

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

7. Open a web browser and enter the following URL to access the application:
http://*<address_of_exposed_route>*/*<java_application_name>*



NOTE

In the preceding example, replace ***<address_of_exposed_route>*** and ***<java_application_name>*** with appropriate values for your deployment.

2.7. ADDING ADDITIONAL JAR FILES IN THE TOMCAT/LIB DIRECTORY

You can use Docker to add additional Java Archive (JAR) files in the **tomcat/lib** directory.

Procedure

1. Start the image in Docker:

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

2. Find the **CONTAINER ID**:

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

3. Copy the library to the **tomcat/lib/** directory:

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

4. Commit the changes to a new image:

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

5. Create a new image tag:

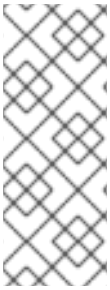
```
docker tag <NEW IMAGE NAME>:latest <NEW IMAGE REGISTRY URL>:<TAG>
```

6. Push the image to a registry:

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

CHAPTER 3. RED HAT JBOSS WEB SERVER METERING LABELS FOR RED HAT OPENSHIFT

You can add metering labels to your Red Hat JBoss Web Server pods and check Red Hat subscription details with the OpenShift Metering Operator.



NOTE

- Do not add metering labels to any pods that an operator or a template deploys and manages.
- You can apply labels to pods using the Metering Operator on OpenShift Container Platform version 4.8 and earlier. From version 4.9 onward, the Metering Operator is no longer available without a direct replacement.

Red Hat JBoss Web Server can use the following metering 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**

Additional resources

- Configuring and using Metering in OpenShift Container Platform

APPENDIX A. S2I SCRIPTS AND MAVEN

The Red Hat JBoss Web Server for OpenShift image includes S2I scripts and Maven.

A.1. MAVEN ARTIFACT REPOSITORY MIRRORS AND JWS FOR OPENSHIFT

A Maven repository holds build artifacts and dependencies, such as the project Java archive (JAR) files, library JAR files, plugins or any other project-specific artifacts. A Maven repository also defines locations that you can download artifacts from while performing the source-to-image (S2I) build. In addition to using the Maven Central Repository, some organizations also deploy a local custom repository (mirror).

A local mirror provides the following benefits:

- Availability of a synchronized mirror that is geographically closer and faster
- Greater control over the repository content
- Possibility to share artifacts across different teams (developers and continuous integration (CI)) without relying on public servers and repositories
- Improved build times

A Maven repository manager can serve as local cache to a mirror. If the repository manager is already deployed and can be reached externally at a specified URL location, the S2I build can use this repository. You can use an internal Maven repository by adding the **MAVEN_MIRROR_URL** environment variable to the build configuration of the application.

Additional resources

- Apache Maven Project: Best Practice - Using a Repository Manager

A.1.1. Using an internal Maven repository for a new build configuration

You can add the **MAVEN_MIRROR_URL** environment variable to a new build configuration of your application, by specifying the **--build-env** option with the **oc new-app** command or the **oc new-build** command.

Procedure

1. Enter the following command:

```
$ oc new-app \
 https://github.com/web-servers/tomcat-websocket-chat-quickstart.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
```

NOTE

The preceding command assumes that the repository manager is already deployed and can be reached at **http://10.0.0.1:8080/repository/internal/**.

A.1.2. Using an internal Maven repository for an existing build configuration

You can add the **MAVEN_MIRROR_URL** environment variable to an existing build configuration of your application, by specifying the name of the build configuration with the **oc set env** command.

Procedure

1. Identify the build configuration that requires the **MAVEN_MIRROR_URL** variable:

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

The preceding command produces the following type of output:

```
buildconfig/jws
```

NOTE

In the preceding example, jws is the name of the build configuration.

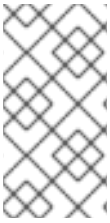
2. Add the **MAVEN_MIRROR_URL** environment variable to **buildconfig/jws**:

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

buildconfig "jws" updated
```

3. Verify the build configuration has updated:

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

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

4. Schedule a new build of the application by using **oc start-build**



NOTE

During the application build process, Maven dependencies are downloaded from the repository manager rather than from the default public repositories. When the build process is completed, the mirror contains all the dependencies that are retrieved and used during the build process.

A.2. SCRIPTS INCLUDED ON THE RED HAT JBOSS WEB SERVER FOR OPENSHIFT IMAGE

The Red Hat JBoss Web Server for OpenShift image includes scripts to run Catalina and to use Maven to create and deploy the **.war** package.

run

Runs Catalina (Tomcat)

assemble

Uses Maven to build the web application source, create the **.war** file, and move the **.war** file to the **$JWS_HOME/tomcat/webapps** directory.

A.3. JWS FOR OPENSHIFT DATASOURCES

JWS for OpenShift provides three type of data sources:

Default internal data sources

PostgreSQL, MySQL, and MongoDB data sources are available on OpenShift by default through the Red Hat Registry. These data sources do not require additional environment files to be configured for image streams. To enable a database to be discovered and used as a data source, you can set the **DB_SERVICE_PREFIX_MAPPING** environment variable to the name of the OpenShift service.

Other internal data sources

These data sources are run on OpenShift but they are not available by default through the Red Hat Registry. Environment files that are added to OpenShift Secrets provide configuration of other internal data sources.

External data sources

These data sources are not run on OpenShift. Environment files that are added to OpenShift Secrets provide configuration of external data sources.

ENV_FILES property

You can add the environment variables for data sources to the OpenShift Secret for the project. You can use the **ENV_FILES** property to call these environment files within the template.

DB_SERVICE_PREFIX_MAPPING environment variable

Data sources are automatically created based on the value of certain environment variables. The **DB_SERVICE_PREFIX_MAPPING** environment variable defines JNDI mappings for the data sources.

The allowed value for the **DB_SERVICE_PREFIX_MAPPING** variable is a comma-separated list of **POOLNAME-DATABASETYPE=PREFIX** triplets. Each triplet consists of the following values:

- **POOLNAME** is used as the **pool-name** in the data source.
- **DATABASETYPE** is the database driver to use.
- **PREFIX** is the prefix in the names of environment variables that are used to configure the data source.

For each **POOLNAME-DATABASETYPE=PREFIX** triplet that is defined in the **DB_SERVICE_PREFIX_MAPPING** environment variable, the launch script creates a separate data source, which is executed when running the image.

Additional resources

- Datasource configuration environment variables

A.4. JWS FOR OPENSHIFT COMPATIBLE ENVIRONMENT VARIABLES

You can modify the build configuration by including environment variables with the source-to-image (S2I) **build** command. For more information, see Maven artifact repository mirrors and JWS for OpenShift.

The following table lists the valid environment variables for the Red Hat JBoss Web Server for OpenShift images:

Variable Name	Display Name	Description	Example Value
ARTIFACT_DIR	N/A	**.war**, **.ear**, and **.jar** files from this directory will be copied into the **deployments** directory	target
APPLICATION_NAME	Application Name	The name for the application	jws-app
CONTEXT_DIR	Context Directory	Path within Git project to build; empty for root project directory	tomcat-websocket-chat
GITHUB_WEBHOOK_SECRET	Github Webhook Secret	Github trigger secret	Expression from: [a-zA-Z0-9]{8}
GENERIC_WEBHOOK_SECRET	Generic Webhook Secret	Generic build trigger secret	Expression from: [a-zA-Z0-9]{8}
HOSTNAME_HTTP	Custom HTTP Route Hostname	Custom hostname for http service route. Leave blank for default hostname	<application-name>-<project>.<default-domain-suffix>
HOSTNAME_HTTPS	Custom HTTPS Route Hostname	Custom hostname for https service route. Leave blank for default hostname	<application-name>-<project>.<default-domain-suffix>
IMAGE_STREAM_NAMESPACE	Imagestream Namespace	Namespace in which the ImageStreams for Red Hat Middleware images are installed	openshift
JWS_HTTPS_CERTIFICATE	Certificate File Name	The name of the certificate file	rsa-cert.pem
JWS_HTTPS_CERTIFICATE_CHAIN	Certificate Chain File Name	The name of the certificate chain file	ca-chain.cert.pem
JWS_HTTPS_CERTIFICATE_DIR	Certificate Directory Name	The name of a directory where the certificate is stored	cert
JWS_HTTPS_CERTIFICATE_KEY	Certificate Key File Name	The name of the certificate key file	rsa-key.pem

Variable Name	Display Name	Description	Example Value
JWS_HTTPS_CERTIFICATE_PASSWORD	Certificate Password	The Certificate Password	P5ssw0rd
SOURCE_REPOSITORY_URL	Git Repository URL	Git source URI for Application	*https://github.com/web-servers/tomcat-websocket-chat-quickstart.git*
SOURCE_REPOSITORY_REFERENCE	Git Reference	Git branch/tag reference	1.2
IMAGE_STREAM_NAMESPACE	Imagestream Namespace	Namespace in which the ImageStreams for Red Hat Middleware images are installed	openshift
MAVEN_MIRROR_URL	Maven Mirror URL	URL of a Maven mirror/repository manager to configure.	http://10.0.0.1:8080/repository/internal/

APPENDIX B. VALVES ON JWS FOR OPENSHIFT

You can define the following environment variable to insert the valve component into the request processing pipeline for the associated Catalina container.

Variable Name	Description	Example Value	Default Value
ENABLE_ACCESS_LOG	Enable the Access Log Valve to log access messages to the standard output channel.	***true***	***false***

APPENDIX C. CHECKING OPENSHIFT LOGS

You can use the **oc logs** command to view the OpenShift logs or the logs that the console provides for a running container.

Procedure

- Enter the following command:

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

NOTE

In the preceding command, replace ***<pod_name>*** and ***<container_name>*** with appropriate values for your deployment.

Access logs are stored in the **/opt/jws-6.0/tomcat/logs/** directory.