



# **Red Hat JBoss Web Server 3.1**

## **Red Hat JBoss Web Server for OpenShift**

Installing and using Red Hat JBoss Web Server for OpenShift



# Red Hat JBoss Web Server 3.1 Red Hat JBoss Web Server for OpenShift

---

Installing and using Red Hat JBoss Web Server for OpenShift

## Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

Guide to using Red Hat JBoss Web Server for OpenShift

---

## Table of Contents

<b>CHAPTER 1. INTRODUCTION</b> .....	<b>3</b>
1.1. WHAT IS RED HAT JBOSS WEB SERVER FOR OPENSIFT?	3
<b>CHAPTER 2. BEFORE YOU BEGIN</b> .....	<b>4</b>
2.1. THE DIFFERENCE BETWEEN RED HAT JBOSS WEB SERVER AND JWS FOR OPENSIFT	4
2.2. VERSION COMPATIBILITY AND SUPPORT	4
<b>CHAPTER 3. GET STARTED</b> .....	<b>5</b>
3.1. INITIAL SETUP	5
3.2. USING THE JWS FOR OPENSIFT SOURCE-TO-IMAGE (S2I) PROCESS	5
3.2.1. Create a JWS for OpenShift application using existing maven binaries	6
3.2.2. Example: Creating a JWS for OpenShift application using existing maven binaries	7
3.2.2.1. Prerequisites:	7
3.2.2.2. To setup the example application on OpenShift	8
3.2.3. Create a JWS for OpenShift application from source code	11
<b>CHAPTER 4. REFERENCE</b> .....	<b>13</b>
4.1. SOURCE-TO-IMAGE (S2I)	13
4.1.1. Using maven artifact repository mirrors with JWS for OpenShift	13
4.1.2. Scripts included on the Red Hat JBoss Web Server for OpenShift image	14
4.1.3. JWS for OpenShift compatible environment variables	14
4.2. VALVES ON JWS FOR OPENSIFT	15
4.2.1. JWS for OpenShift compatible environmental variables (valve component)	15
4.3. CHECKING LOGS	15



## CHAPTER 1. INTRODUCTION

### 1.1. WHAT IS RED HAT JBOSS WEB SERVER FOR OPENSIFT?

The Apache Tomcat 7 and Apache Tomcat 8 components of Red Hat JBoss Web Server (JWS) 3.1 are available as containerized images designed for OpenShift. Developers can use these images to build, scale, and test Java web applications for deployment across hybrid cloud environments.

## CHAPTER 2. BEFORE YOU BEGIN

### 2.1. THE DIFFERENCE BETWEEN RED HAT JBOSS WEB SERVER AND JWS FOR OPENSIFT

The differences between the JWS for OpenShift images and the regular release of JWS are:

- The location of ***JWS\_HOME/tomcat<version>/*** inside a JWS for OpenShift image is: ***/opt/webserver/***.
- The JWS for OpenShift images do not contain Apache HTTP Server. All load balancing is handled by the OpenShift router, not Apache HTTP Server `mod_cluster` or `mod_jk` connectors.

Documentation for JWS functionality not specific to JWS for OpenShift images is found in the [Red Hat JBoss Web Server documentation](#).

### 2.2. VERSION COMPATIBILITY AND SUPPORT

See the xPaaS table on the [OpenShift Container Platform Tested Integrations page](#) for details about OpenShift image version compatibility.



#### IMPORTANT

**The 3.1 version of JWS for OpenShift images and application templates should be used for deploying new applications.**

The 3.0 version of JWS for OpenShift images and application templates are deprecated and no longer receives updates.

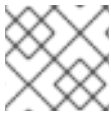


## CHAPTER 3. GET STARTED

### 3.1. INITIAL SETUP

The instructions in this guide follow on from the [OpenShift Primer](#), assuming a supported OpenShift configuration or a non-production OpenShift instance like that described in the OpenShift Primer.

The JWS for OpenShift images are [automatically created during the installation](#) of OpenShift, along with the other default image streams and templates.



#### NOTE

The JWS for OpenShift application templates are distributed for Tomcat 7, and Tomcat 8.

### 3.2. USING THE JWS FOR OPENSIFT SOURCE-TO-IMAGE (S2I) PROCESS

To run and configure the JWS for OpenShift images, use the OpenShift S2I process with the application template parameters and environment variables.

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

- If there is a Maven **settings.xml** file in the **configuration/** source directory, it is moved to **\$HOME/.m2/** of the new image.  
See the [Apache Maven Project website](#) for more information on Maven and the Maven **settings.xml** file.
- If there is a **pom.xml** file in the source repository, 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, including the arguments for skipping tests (**-DskipTests**) and enabling the Red Hat GA repository (**-Dcom.redhat.xpaas.repo.redhatga**).
- The results of a successful Maven build are copied to **/opt/webserver/webapps/**. This includes all WAR files from the source directory specified by the **\$ARTIFACT\_DIR** environment variable. The default value of **\$ARTIFACT\_DIR** is the **target/** directory.  
Use the **MAVEN\_ARGS\_APPEND** environment variable to modify the Maven arguments.
- All WAR files from the **deployments/** source directory are copied to **/opt/webserver/webapps/**.
- All files in the **configuration/** source directory are copied to **/opt/webserver/conf/** (excluding the Maven **settings.xml** file).
- All files in the **lib/** source directory are copied to **/opt/webserver/lib/**.



#### NOTE

If you want to use custom Tomcat configuration files, the file names should be the same as for a normal Tomcat installation. For example, **context.xml** and **server.xml**.

See the [Artifact Repository Mirrors](#) section for guidance on configuring the S2I process to use a custom Maven artifacts repository mirror.

### 3.2.1. Create a JWS for OpenShift application using existing maven binaries

Existing applications are deployed on OpenShift using the `oc start-build` command.

*Prerequisite:* An existing `.war`, `.ear`, or `.jar` of the application to deploy on JWS for OpenShift.

1. Prepare the directory structure on the local file system.  
Create a source directory containing any content required by your application not included in the binary (if required, see [Using the JWS for OpenShift Source-to-Image \(S2I\) process](#)), then create a subdirectory `deployments/`:

```
$ mkdir -p <build_dir>/deployments
```

2. Copy the binaries (`.war`, `.ear`, `.jar`) to `deployments/`:

```
$ cp /path/to/binary/<filenames_with_extensions>  
<build_dir>/deployments/
```



#### NOTE

Application archives in the `deployments/` subdirectory of the source directory are copied to the `$JWS_HOME/webapps/` directory of the image being built on OpenShift. For the application to deploy, the directory hierarchy containing the web application data must be structured correctly (see [Section 3.2, “Using the JWS for OpenShift Source-to-Image \(S2I\) process”](#)).

3. Log in to the OpenShift instance:

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

4. Create a new project if required:

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

5. Identify the JWS for OpenShift image stream to use for your application with `oc get is -n openshift`:

```
$ oc get is -n openshift | grep ^jboss-webserver | cut -f1 -d ' '  
  
jboss-webserver30-tomcat7-openshift  
jboss-webserver30-tomcat8-openshift  
jboss-webserver31-tomcat7-openshift  
jboss-webserver31-tomcat8-openshift
```



#### NOTE

The option `-n openshift` specifies the project to use. `oc get is -n openshift` retrieves (`get`) the image stream resources (`is`) from the `openshift` project.

6. Create the new build configuration, specifying image stream and application name:

```
$ oc new-build --binary=true \
  --image-stream=jboss-webserver31-tomcat8-openshift \
  --name=<my-jws-on-openshift-app>
```

7. Instruct OpenShift to use the source directory created [previously](#) for binary input of the OpenShift image build:

```
$ oc start-build <my-jws-on-openshift-app> --from-dir=./<build_dir>
--follow
```

8. Create a new OpenShift application based on the image:

```
$ oc new-app <my-jws-on-openshift-app>
```

9. Expose the service to make the application accessible to users:

```
# to check the name of the service to expose
$ oc get svc -o name

service/<my-jws-on-openshift-app>

# to expose the service
$ oc expose svc/my-jws-on-openshift-app

route "my-jws-on-openshift-app" exposed
```

10. Retrieve the address of the exposed route:

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

11. To access the application in your browser: **`http://<address_of_exposed_route> / <my-war-ear-jar-filename-without-extension>`**

## 3.2.2. Example: Creating a JWS for OpenShift application using existing maven binaries

The example below uses the [tomcat-websocket-chat](#) quickstart using the procedure from [Section 3.2.1](#), “Create a JWS for OpenShift application using existing maven binaries”.

### 3.2.2.1. Prerequisites:

- A. Get the WAR application archive or build the application locally.

- Clone the source code:

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

- [Configure the Red Hat JBoss Middleware Maven repository](#),

- Build the application:

```
$ cd openshift-quickstarts/tomcat-websocket-chat/

$ mvn clean package

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

#### B. Prepare the directory structure on the local file system.

Create the source directory for the binary build on your local file system and the **deployments/** subdirectory. Copy the WAR archive to **deployments/**:

```
[tomcat-websocket-chat]$ ls

pom.xml  README.md  src/  target/

$ mkdir -p ocp/deployments

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

#### 3.2.2.2. To setup the example application on OpenShift

1. Log in to the OpenShift instance:

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

2. Create a new project if required:

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

3. Identify the JWS for OpenShift image stream to use for your application with **oc get is -n openshift**:

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

```

jboss-webserver30-tomcat7-openshift
jboss-webserver30-tomcat8-openshift
jboss-webserver31-tomcat7-openshift
jboss-webserver31-tomcat8-openshift

```

4. Create new build configuration, specifying image stream and application name:

```

$ oc new-build --binary=true \
  --image-stream=jboss-webserver31-tomcat8-openshift \
  --name=jws-wsch-app

--> Found image 8c3b85b (4 weeks old) in image stream
"openshift/jboss-webserver31-tomcat8-openshift" under tag "latest"
for "jboss-webserver31-tomcat8-openshift"

      JBoss Web Server 3.1
      -----
      Platform for building and running web applications on JBoss Web
      Server 3.1 - Tomcat v8

      Tags: builder, java, tomcat8

      * 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

```

5. Start the binary build. Instruct OpenShift to use source directory for the binary input for the OpenShift image build:

```

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

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/webserver/webapps
for later deployment...
'/home/jboss/source/deployments/websocket-chat.war' ->
'/opt/webserver/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
```

6. Create a new OpenShift application based on the image:

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

--> 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 3.1
-----
Platform for building and running web applications on JBoss Web
Server 3.1 - Tomcat v8

Tags: builder, java, tomcat8

* 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.
```

7. Expose the service to make the application accessible to users:

```
# to check the name of the service to expose
$ oc get svc -o name

service/jws-wsch-app

# to expose the service
$ oc expose svc/jws-wsch-app

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

8. Retrieve the address of the exposed route:

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

9. Access the application in your browser: [http://<address\\_of\\_exposed\\_route>/websocket-chat](http://<address_of_exposed_route>/websocket-chat)

### 3.2.3. Create a JWS for OpenShift application from source code

For detailed instructions on creating new OpenShift applications from source code, see [OpenShift.com - Creating an Application From Source Code](#).



#### NOTE

Before proceeding, ensure that the applications' data is structured correctly (see [Section 3.2, “Using the JWS for OpenShift Source-to-Image \(S2I\) process”](#)).

1. Log in to the OpenShift instance:

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

2. Create a new project if required:

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

3. Identify the JWS for OpenShift image stream to use for your application with `oc get is -n openshift`:

```
$ oc get is -n openshift | grep ^jboss-webserver | cut -f1 -d ' '
jboss-webserver30-tomcat7-openshift
jboss-webserver30-tomcat8-openshift
jboss-webserver31-tomcat7-openshift
jboss-webserver31-tomcat8-openshift
```

4. Create the new OpenShift application from source code using Red Hat JBoss Web Server for OpenShift images, use the `--image-stream` option:

```
$ oc new-app \
  <source_code_location> \
  --image-stream=jboss-webserver31-tomcat8-openshift \
  --name=<openshift_application_name>
```

For Example:

```
$ oc new-app \
  https://github.com/jboss-openshift/openshift-quickstarts.git#master \
  --image-stream=jboss-webserver31-tomcat8-openshift \
  --context-dir='tomcat-websocket-chat' \
  --name=jws-wsch-app
```

The source code is added to the image and the source code is compiled. The build configuration and services are also created.

5. To expose the application:

```
# to check the name of the service to expose
$ oc get svc -o name
```

```
service/<openshift_application_name>  
  
# to expose the service  
$ oc expose svc/<openshift_application_name>  
  
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. To access the application in your browser:

[http://<address\\_of\\_exposed\\_route>/<java\\_application\\_name>](http://<address_of_exposed_route>/<java_application_name>)



## CHAPTER 4. REFERENCE

### 4.1. SOURCE-TO-IMAGE (S2I)

The Red Hat JBoss Web Server for OpenShift image includes [S2I scripts](#) and Maven.

#### 4.1.1. Using maven artifact repository mirrors with JWS for OpenShift

A Maven repository holds build artifacts and dependencies, such as the project jars, library jars, plugins or any other project specific artifacts. It also defines locations to download artifacts from while performing the S2I build. Along with using the [Maven Central Repository](#), some organizations also deploy a local custom repository (mirror).

Benefits of using a local mirror are:

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

A [Maven repository manager](#) can serve as local cache to a mirror. Assuming that the repository manager is already deployed and reachable externally at `http://10.0.0.1:8080/repository/internal/`, the S2I build can use this repository. To use an internal Maven repository, add the `MAVEN_MIRROR_URL` environment variable to the build configuration of the application.

For a new build configuration, use the `--build-env` option with `oc new-app` or `oc new-build`:

```
$ oc new-app \
  https://github.com/jboss-openshift/openshift-quickstarts.git#master \
  --image-stream=jboss-webserver31-tomcat8-openshift \
  --context-dir='tomcat-websocket-chat' \
  --build-env MAVEN_MIRROR_URL=http://10.0.0.1:8080/repository/internal/ \
  --name=jws-wsch-app
```

For an existing build configuration:

1. Identify the build configuration which requires the `MAVEN_MIRROR_URL` variable:

```
$ oc get bc -o name
buildconfig/jws
```

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

```
$ oc 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 env bc/jws --list
# buildconfigs jws
MAVEN_MIRROR_URL=http://10.0.0.1:8080/repository/internal/
```

- Schedule a new build of the application using `oc start-build`



#### NOTE

During application build, Maven dependencies are download from the repository manager, instead of the default public repositories. Once the build has finished, the mirror contains all the dependencies retrieved and used during the build.

### 4.1.2. Scripts included on the Red Hat JBoss Web Server for OpenShift image

#### run

runs Catalina (Tomcat)

#### assemble

uses Maven to build the source, create package (`.war`) and move it to the `$JWS_HOME/webapps` directory.

### 4.1.3. JWS for OpenShift compatible environment variables

The build configuration can be modified by including environment variables to the Source-to-Image `build` command (see [Section 4.1.1, “Using maven artifact repository mirrors with JWS for OpenShift”](#)). The valid environment variables for the Red Hat JBoss Web Server for OpenShift images are:

Variable Name	Description	Example Value :leveloffset: +3
<b>ARTIFACT_DIR</b>	<code>.war</code> , <code>.ear</code> , and <code>.jar</code> files from this directory will be copied into the <b>deployments</b> directory.	<i>target</i>
<b>HTTP_PROXY_HOST</b>	Hostname or IP address of a HTTP proxy for Maven to use.	<i>192.168.1.1</i>
<b>HTTP_PROXY_PORT</b>	TCP Port of a HTTP proxy for Maven to use.	<i>8080</i>
<b>HTTP_PROXY_USERNAME</b>	If supplied with <b>HTTP_PROXY_PASSWORD</b> , use credentials for HTTP proxy.	<i>myusername</i>
<b>HTTP_PROXY_PASSWORD</b>	If supplied with <b>HTTP_PROXY_USERNAME</b> , use credentials for HTTP proxy.	<i>mypassword</i>

Variable Name	Description	Example Value :leveloffset: +3
<b><i>HTTP_PROXY_NONPROXYHOSTS</i></b>	If supplied, a configured HTTP proxy will ignore these hosts (a comma-separated lists of hosts, IP addresses or domains).	<b><i>*.example.net,some.example.org</i></b>
<b><i>MAVEN_ARGS</i></b>	Overrides the arguments supplied to Maven during build.	<b><i>-e -Popenshift -DskipTests -Dcom.redhat.xpaas.repo.redhatga package</i></b>
<b><i>MAVEN_ARGS_APPEND</i></b>	Appends user arguments supplied to Maven during build.	<b><i>-Dfoo=bar</i></b>
<b><i>MAVEN_MIRROR_URL</i></b>	URL of a Maven mirror/repository manager to configure.	<b><i>http://10.0.0.1:8080/repository/internal/</i></b>
<b><i>MAVEN_CLEAR_REPO</i></b>	Optionally clear the local Maven repository after the build.	<b><i>true</i></b>  :leveloffset: 3

## 4.2. VALVES ON JWS FOR OPENSIFT

### 4.2.1. JWS for OpenShift compatible environmental variables (valve component)

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

Variable Name	Description	Example Value	Default Value
<b><i>ENABLE_ACCESS_LOG</i></b>	Enable the Access Log Valve to log access messages to the standard output channel.	<b><i>true</i></b>	<b><i>false</i></b>

## 4.3. CHECKING LOGS

To view the OpenShift logs or the logs provided by a running container's console:

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

Access logs are stored in ***/opt/webserver/logs/***.