



Red Hat JBoss Web Server 5.0

Red Hat JBoss Web Server for OpenShift

Red Hat JBoss Web Server for OpenShift のインストールおよび使用

Red Hat JBoss Web Server 5.0 Red Hat JBoss Web Server for OpenShift

Red Hat JBoss Web Server for OpenShift のインストールおよび使用

法律上の通知

Copyright © 2023 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, the Red Hat 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 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.

概要

Red Hat JBoss Web Server for OpenShift の使用ガイド

目次

第1章 はじめに	3
1.1. RED HAT JBOSS WEB SERVER FOR OPENSIFT とは	3
第2章 操作を始める前に	4
2.1. RED HAT JBOSS WEB SERVER と JWS FOR OPENSIFT の相違点	4
2.2. バージョンの互換性とサポート	4
第3章 使い始める	5
3.1. 初期設定	5
3.2. OPENSIFT SOURCE-TO-IMAGE (S2I) プロセスでの JWS の使用	5
第4章 リファレンス	12
4.1. SOURCE-TO-IMAGE (S2I)	12
4.2. OPENSIFT 用の JWS 上のバルブ	14
4.3. ログの確認	14

第1章 はじめに

1.1. RED HAT JBOSS WEB SERVER FOR OPENSIFT とは

Red Hat JBoss Web Server (JWS) 5.0 の Apache Tomcat 7 および Apache Tomcat 8 コンポーネントは、OpenShift 用に設計されたコンテナ化イメージとして提供しています。開発者は、これらのイメージを使用して、ハイブリッドクラウド環境全体にデプロイメントする Java Web アプリケーションを構築、スケーリング、およびテストできます。

第2章 操作を始める前に

2.1. RED HAT JBOSS WEB SERVER と JWS FOR OPENSIFT の相違点

OpenShift イメージの JWS と JWS の定期的なリリースの違いは次のとおりです。

- JWS for OpenShift イメージでの **JWS_HOME/tomcat<version>/** は **/opt/webserver/** にあります。
- OpenShift イメージ用の JWS には Apache HTTP Server が含まれていません。すべての負荷分散は、Apache HTTP Server の mod_cluster または mod_jk コネクターではなく、OpenShift ルーターによって処理されます。

JWS for OpenShift イメージに固有の JWS 機能のドキュメントは、[Red Hat JBoss Web Server のドキュメント](#) を参照してください。

2.2. バージョンの互換性とサポート

OpenShift イメージバージョンの互換性についての詳細は、[OpenShift Container Platform Tested Integrations ページ](#) の xPaaS テーブルを参照してください。



重要

新しいアプリケーションのデプロイには、OpenShift イメージおよびアプリケーションテンプレート用の JWS の 5.0 バージョンを使用する必要があります。

JWS for OpenShift イメージおよびアプリケーションテンプレートの 3.1 バージョンは非推奨となり、更新を受信しなくなりました。

第3章 使い始める

3.1. 初期設定

本書の手順は [OpenShift Primer](#) に従います。OpenShift Primer に記述されているサポートされる OpenShift の設定または非実稼働環境の OpenShift インスタンスを想定しています。

JWS for OpenShift イメージは、OpenShift の [インストール時に、他のデフォルトイメージストリームおよびテンプレートと共に自動的に作成されます](#)。



注記

JWS for OpenShift アプリケーションテンプレートは、Tomcat 7 および Tomcat 8 用に配布されています。

3.2. OPENSIFT SOURCE-TO-IMAGE (S2I) プロセスでの JWS の使用

JWS for OpenShift イメージを実行および設定するには、アプリケーションテンプレートパラメーターおよび環境変数を使用して OpenShift S2I プロセスを使用します。

JWS for OpenShift イメージの S2I プロセスは以下のようになります。

- **configuration/** Maven **settings.xml** ファイルがある場合は、新しいイメージの **\$HOME/.m2/** に移動します。
Maven および Maven の **settings.xml** ファイルの詳細は、[Apache Maven Project の Web サイト](#) を参照してください。
- ソースリポジトリに **pom.xml** ファイルがある場合、**\$MAVEN_ARGS** 環境変数の内容を使用して Maven ビルドがトリガーされます。
デフォルトでは、**package** のゴールは **openshift** プロファイルで使用されます。これには、テストをスキップする引数 (**-DskipTests**) および Red Hat GA リポジトリを有効にするための引数 (**-Dcom.redhat.xpaas.repo.redhatga**) が含まれます。
- 成功した Maven ビルドの結果は **/opt/webserver/webapps/** にコピーされます。これには、**\$ARTIFACT_DIR** 環境変数によって指定されたソースディレクトリーからの WAR ファイルがすべて含まれます。**\$ARTIFACT_DIR** のデフォルト値は **target/** ディレクトリーです。
MAVEN_ARGS_APPEND 環境変数を使用して Maven 引数を変更します。
- **deployments/** ソースディレクトリーのすべての WAR ファイルが **/opt/webserver/webapps/** にコピーされます。
- **configuration/** ソースディレクトリーのすべてのファイルは、**/opt/webserver/conf/** にコピーされます (Maven の **settings.xml** ファイルを除く)。
- **lib/** ソースディレクトリーのすべてのファイルは **/opt/webserver/lib/** にコピーされます。



注記

カスタムの Tomcat 設定ファイルを使用する場合は、通常の Tomcat インストールと同じファイル名を使用する必要があります。例: **context.xml** および **server.xml**。

S2I プロセスを設定してカスタム Maven アーティファクトリーポジトリミラーを使用する方法は、[アーティファクトリーポジトリミラー](#) の項を参照してください。

3.2.1. 既存の Maven バイナリーを使用した OpenShift アプリケーションの JWS の作成

既存のアプリケーションは、**oc start-build** コマンドを使用して OpenShift にデプロイされます。

前提条件: OpenShift 用の JWS にデプロイする既存の **.war**、**.ear**、または **.jar** アプリケーション。

- ローカルファイルシステムでディレクトリー構造を準備します。
アプリケーションが必要とするコンテンツがバイナリーに含まれていないコンテンツを含むソースディレクトリーを作成します。必要に応じて、[Using the JWS for OpenShift Source-to-Image\(S2I\) プロセス](#) を参照してください。次に、**deployments/** サブディレクトリーを作成します。

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

- バイナリー (**.war**、**.ear**、**.jar**) を **deployments/** にコピーします。

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



注記

ソースディレクトリーの **deployments/** サブディレクトリーにあるアプリケーションアーカイブは、OpenShift 上にビルドされているイメージの **\$JWS_HOME/webapps/** ディレクトリーにコピーされます。アプリケーションをデプロイするには、web アプリケーションデータが含まれるディレクトリー階層が正しく設定される必要があります (「[OpenShift Source-to-Image \(S2I\) プロセスでの JWS の使用](#)」を参照)。

- OpenShift インスタンスにログインします。

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

- 必要に応じて新規プロジェクトを作成します。

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

- oc get is -n openshift** でアプリケーションに使用する OpenShift イメージストリームの JWS を特定します。

```
$ 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
```



注記

オプション **-n openshift** は使用するプロジェクトを指定します。**oc get is -n openshift** は **openshift** プロジェクトからイメージストリームリソース (**is**) を取得 (**get**) します。

- イメージストリームおよびアプリケーション名を指定して、新規ビルド設定を作成します。

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

7. OpenShift イメージビルドのバイナリー入力用に [以前](#) 作成されたソースディレクトリーを使用するように OpenShift に指示します。

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

8. イメージに基づいて新しい OpenShift アプリケーションを作成します。

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

9. サービスを公開して、アプリケーションがユーザーにアクセスできるようにします。

```
# 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. 公開されたルートのアドレスを取得します。

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

11. ブラウザーでアプリケーションにアクセスするには `http://<address_of_exposed_route>/<my-war-ear-jar-filename-without-extension>` を入力します。

3.2.2. 例: 既存の Maven バイナリーを使用した OpenShift アプリケーションの JWS の作成

以下の例では、「[既存の Maven バイナリーを使用した OpenShift アプリケーションの JWS の作成](#)」の手順を使用して `tomcat-websocket-chat` クイックスタートを使用します。

3.2.2.1. 要件:

- A. WAR アプリケーションアーカイブを取得したり、アプリケーションをローカルにビルドしたりします。

- ソースコードのクローンを作成します。

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

- [Red Hat JBoss Middleware Maven リポジトリ](#) を [設定](#) します。
- アプリケーションをビルドします。

```
$ 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. ローカルファイルシステムでディレクトリー構造を準備します。

バイナリービルドのソースディレクトリーを、ローカルファイルシステムと **deployments/** サブディレクトリーに作成します。WAR アーカイブを **deployments/** にコピーします。

```
[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. OpenShift でサンプルアプリケーションを設定する方法

1. OpenShift インスタンスにログインします。

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

2. 必要に応じて新規プロジェクトを作成します。

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

3. **oc get is -n openshift** でアプリケーションに使用する OpenShift イメージストリームの JWS を特定します。

```
$ 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. イメージストリームおよびアプリケーション名を指定して、新規ビルド設定を作成します。

```
$ 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
```

- バイナリービルドを開始します。OpenShift イメージビルドのバイナリー入力にソースディレクトリーを使用するように OpenShift に指示します。

```
$ 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
```

- イメージに基づいて新しい OpenShift アプリケーションを作成します。

```
$ 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. サービスを公開して、アプリケーションがユーザーにアクセスできるようにします。

```
# 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. 公開されたルートのアドレスを取得します。

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

9. ブラウザーでアプリケーションにアクセスします:
http://<address_of_exposed_route>/websocket-chat

3.2.3. ソースコードから JWS for OpenShift アプリケーションを作成します。

ソースコードから新しい OpenShift アプリケーションを作成する詳細な手順は、[OpenShift.com - ソースコードからのアプリケーションの作成](#) を参照してください。



注記

次に進む前に、アプリケーションのデータが正しく構造化されていることを確認します (「[OpenShift Source-to-Image \(S2I\) プロセスでの JWS の使用](#)」を参照)。

1. OpenShift インスタンスにログインします。

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

2. 必要に応じて新規プロジェクトを作成します。

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

3. **oc get is -n openshift** でアプリケーションに使用する OpenShift イメージストリームの JWS を特定します。

```
$ 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. Red Hat JBoss Web Server for OpenShift イメージを使用して、ソースコードから新しい OpenShift アプリケーションを作成し、**--image-stream** オプションを使用します。

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

例を以下に示します。

```
$ 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
```

ソースコードがイメージに追加され、ソースコードがコンパイルされます。ビルド設定とサービスも作成されます。

5. アプリケーションを公開するには、以下を実行します。

```
# 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. 公開されたルートアドレスを取得するには、以下を実行します。

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

7. ブラウザーでアプリケーションにアクセスします:

```
http://<address_of_exposed_route>/<java_application_name>
```

第4章 リファレンス

4.1. SOURCE-TO-IMAGE (S2I)

Red Hat JBoss Web Server for OpenShift イメージには、[S2I スクリプト](#) と Maven が含まれています。

4.1.1. OpenShift の JWS での Maven アーティファクトリポジトリミラーの使用

Maven リポジトリは、プロジェクト jar、ライブラリー jar、プラグイン、その他のプロジェクト固有のアーティファクトなどのビルドアーティファクトおよび依存関係を保持します。また、S2I ビルドの実行中にアーティファクトをダウンロードするための場所も定義します。[Maven Central Repository](#) の使用に加えて、ローカルのカスタムリポジトリ (mirror) もデプロイします。

ローカルミラーを使用する利点は次のとおりです。

- 地理的に近く、高速な同期ミラーを使用できる。
- リポジトリコンテンツの制御が強化されます。
- パブリックサーバーおよびリポジトリに依存する必要なく、異なるチーム (開発者、CI) 全体でアーティファクトを共有できる。
- ビルド時間が改善される。

[Maven リポジトリマネージャー](#) はミラーへのローカルキャッシュとして機能できます。リポジトリマネージャーがすでにデプロイされ、外部で <http://10.0.0.1:8080/repository/internal/> にアクセスできる場合、S2I ビルドはこのリポジトリを使用できます。内部 Maven リポジトリを使用するには、**MAVEN_MIRROR_URL** 環境変数をアプリケーションのビルド設定に追加します。

新規ビルド設定の場合は、**oc new-app** または **oc new-build** で **--build-env** オプションを指定します。

```
$ 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
```

既存のビルド設定の場合

1. **MAVEN_MIRROR_URL** 変数を必要とするビルド設定を特定します。

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

2. **MAVEN_MIRROR_URL** 環境変数を **buildconfig/jws** に追加します。

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

3. ビルド設定が更新されたことを確認します。


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

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

4. **oc start-build** を使用したアプリケーションの新規ビルドのスケジュール



注記

アプリケーションのビルド中、Maven 依存関係はデフォルトのパブリックリポジトリではなく、リポジトリマネージャーからプルされることを確認できます。ビルドが完了すると、ミラーにはビルド時に取得され、使用されるすべての依存関係が含まれます。

4.1.2. Red Hat JBoss Web Server for OpenShift イメージに含まれるスクリプト

run

Catalina (Tomcat) の実行

assemble

Maven を使用してソースをビルドし、パッケージ (.war) を作成して、**\$JWS_HOME/webapps** ディレクトリに移動します。

4.1.3. OpenShift と互換性のある環境変数の JWS

ビルド設定は、Source-to-Image **build** コマンドに環境変数を含めることで変更できます (「[OpenShift の JWS での Maven アーティファクトリポジトリミラーの使用](#)」を参照)。Red Hat JBoss Web Server for OpenShift イメージに有効な環境変数は以下のとおりです。

変数名	説明	値の例
		:leveloffset: +3
ARTIFACT_DIR	このディレクトリーの .war、.ear、および.jar ファイル は、 deployments ディレクト リーにコピーされます。	target
HTTP_PROXY_HOST	Maven が使用する HTTP プロキ シーのホスト名または IP アドレ ス。	192.168.1.1
HTTP_PROXY_PORT	Maven が使用する HTTP プロキ シーの TCP ポート。	8080
HTTP_PROXY_USERNAME	HTTP_PROXY_PASSWORD と指定された場合、HTTP プロキ シーのクレデンシャルを使用しま す。	myusername

HTTP_PROXY_PASSWORD	HTTP_PROXY_USERNAME と指定された場合、HTTP プロキシのクレデンシャルを使用します。	mypassword
HTTP_PROXY_NONPROXYHOSTS	指定した場合、設定された HTTP プロキシはこれらのホスト (ホスト、IP アドレス、またはドメインのコンマ区切りリスト) を無視します。	*.example.net,some.example.org
MAVEN_ARGS	ビルド中に Maven に指定された引数をオーバーライドします。	-e -Popenshift -DskipTests -Dcom.redhat.xpaas.repo.redhatga package
MAVEN_ARGS_APPEND	ビルド中に Maven に指定されたユーザー引数を追加します。	-Dfoo=bar
MAVEN_MIRROR_URL	設定する Maven ミラー/リポジトリマネージャーの URL。	http://10.0.0.1:8080/repository/internal/
MAVEN_CLEAR_REPO	任意で、ビルド後にローカル Maven リポジトリを消去します。	true :leveloffset: 3

4.2. OPENSIFT 用の JWS 上のバルブ

4.2.1. OpenShift と互換性のある環境変数の JWS (バルブコンポーネント)

以下の環境変数を定義して、バルブコンポーネントに関連付けられた Catalina コンテナのリクエスト処理パイプラインに挿入することができます。

変数名	説明	値の例	デフォルト値
ENABLE_ACCESS_LOG	Access Log Valve を有効にして、標準出力チャンネルへのアクセスメッセージをログに記録します。	true	false

4.3. ログの確認

実行中のコンテナのコンソールが提供する OpenShift ログまたはログを表示するには、以下を実行します。

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

アクセスログは `/opt/webserver/logs/` に保存されます。

