



Red Hat xPaaS

0

Red Hat xPaaS EAP Image

Guide to developing with the Red Hat Enterprise Application Platform Image

Red Hat xPaaS Documentation
Team

Guide to developing with the Red Hat Enterprise Application Platform Image

Legal Notice

Copyright © 2016 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 the Red Hat xPaaS EAP image

Table of Contents

CHAPTER 1. INTRODUCTION TO THE XPAAS EAP IMAGE	3
1.1. WHAT IS JBOSS ENTERPRISE APPLICATION PLATFORM (EAP)?	3
1.2. HOW DOES JBOSS EAP WORK ON OPENSIFT?	3
1.3. COMPARISON: EAP AND XPAAS EAP IMAGE	3
1.4. COMPARISON: EAP 6.4 AND XPAAS EAP 7.0 IMAGE	4
1.5. VERSION COMPATIBILITY AND SUPPORT	4
CHAPTER 2. INSTALLATION AND CONFIGURATION	5
2.1. OVERVIEW	5
2.2. KEY TERMS	5
2.3. INITIAL SETUP	6
2.4. GETTING STARTED	6
2.5. CONFIGURING THE XPAAS EAP IMAGE	7
CHAPTER 3. REFERENCE INFORMATION	9
3.1. DATASOURCES	9
3.2. CLUSTERING	11
3.3. SECURITY DOMAINS	12
3.4. HTTPS	13
3.5. ADMINISTRATION	14
3.6. S2I	14
3.7. SSO	16

CHAPTER 1. INTRODUCTION TO THE XPAAS EAP IMAGE

1.1. WHAT IS JBOSS ENTERPRISE APPLICATION PLATFORM (EAP)?

Red Hat JBoss Enterprise Application Platform 7.0 (JBoss EAP 7) is an application server that works as a middleware platform, is built on open standards, and is compliant with the Java EE 7 specification.

It is based on Wildfly 10, and provides preconfigured options for features such as high-availability clustering, messaging, and distributed caching.

With JBoss EAP you can develop, deploy, and run applications using the various APIs and services that JBoss EAP provides. JBoss EAP includes a modular structure that allows you to enable services only when required, which results in improved startup speed. The web-based management console and management command line interface (CLI) make editing XML configuration files unnecessary and add the ability to script and automate tasks. In addition, JBoss EAP includes APIs and development frameworks for quickly developing secure and scalable Java EE applications. JBoss EAP 7 is a certified implementation of the Java EE 7 full and web profile specifications.

1.2. HOW DOES JBOSS EAP WORK ON OPENSIFT?

Red Hat offers a containerized xPaaS image for the Red Hat JBoss Enterprise Application Platform (JBoss EAP) that is designed for use with OpenShift. Using this image, developers can quickly and easily build, scale, and test applications that are deployed across hybrid environments.

1.3. COMPARISON: EAP AND XPAAS EAP IMAGE

There are some notable differences when comparing the JBoss EAP product with the available JBoss EAP xPaaS image. The following table describes these differences and notes which features are included or supported in the current version of the EAP xPaaS image.

Table 1.1. Differences between EAP and EAP xPaaS Image

Feature	Status	Description
EAP Management Console	Not included	The EAP Management Console is not included in this release of EAP xPaaS image.
Domain mode	Not supported	Although domain mode is not supported, creation and distribution of applications are managed in the containers on OpenShift.
Default root page	Disabled	The default root page is disabled, but you can deploy your own application to the root context as ROOT.war .

Feature	Status	Description
Remote messaging	Supported	A-MQ for inter-pod and remote messaging is supported. HornetQ is only supported for intra-pod messaging and only enabled when A-MQ is absent. The EAP 7 xPaaS image includes Artemis as a replacement for HornetQ.

1.4. COMPARISON: EAP 6.4 AND XPAAS EAP 7.0 IMAGE

Red Hat offers two xPaaS EAP images for use with OpenShift. The first is based on JBoss EAP 6.4 and the second is based on JBoss EAP 7. There are several differences between the two images:

JBoss Web is replaced by Undertow

- ✳ The xPaaS JBoss EAP 6.4 image uses JBoss Web.
- ✳ The xPaaS JBoss EAP 7 image uses Undertow instead of JBoss Web. This change only affects users implementing custom JBoss Web Valves in their applications. Affected users must refer to the Red Hat JBoss EAP 7 documentation for details about [migrating JBoss EAP Web Valve handlers](#).

HornetQ is replaced by Artemis

- ✳ The EAP 6.4 image only uses HornetQ for intra-pod messaging when A-MQ is absent.
- ✳ The EAP 7 image uses Artemis instead of HornetQ. This change resulted in renaming the **HORNETQ_QUEUES** and **HORNETQ_TOPICS** environment variables to **MQ_QUEUES** and **MQ_TOPICS** respectively. For complete instructions to deal with migrating applications from JBoss EAP 6.4 to 7, see the [JBoss EAP 7 Migration Guide](#).

1.5. VERSION COMPATIBILITY AND SUPPORT

The xPaaS images for OpenShift are updated frequently. Therefore, it is important to understand which versions of the xPaaS images are compatible with which versions of OpenShift. Not all images are compatible with all OpenShift 3.x versions. Visit the Red Hat Customer Portal and see the [OpenShift and Atomic Platform Tested Integrations page](#) for more information on version compatibility and support.

CHAPTER 2. INSTALLATION AND CONFIGURATION

2.1. OVERVIEW

Before installing the JBoss EAP xPaaS image on your OpenShift instance, you must first determine whether you are installing the EAP xPaaS image in a production or a non-production environment. Production environments require Secure Sockets Layer (SSL) encryption for network communication for general public access, which is also known as a HTTPS connection. In this case you must use a signed certificate from a Certificate Authority (CA).

However, if you are installing the EAP xPaaS image for demonstration purposes, proof-of-concept (POC) designs, or environments with internal access only, unencrypted and insecure communication may be sufficient. The instructions referenced here describe how to create the required keystore for the xPaaS EAP image with a self-signed or a purchased SSL certificate.

Warning

Using a self-signed SSL certificate to create a keystore is not intended for production environments. For production environments or where SSL encrypted communication is required, you must use a SSL certificate that is purchased from a verified CA.

2.2. KEY TERMS

The following table describes the various terms that are used within the context of this topic.

Table 2.1. Terminology used in this topic

Key term	Description
SSL	Secure Sockets Layer encrypts network traffic between the client and the EAP web server, providing a HTTPS connection between them.
HTTPS	HTTPS is a protocol that provides an SSL-encrypted connection between a client and a server.
Keystore	A Java keystore is a repository to store SSL/TLS certificates and distribute them to applications for encrypted communication.
Secrets	A secret contains the Java keystore that gets passed to the EAP xPaaS image along with a password to access it. This then gets used in scripts to configure HTTPS access.

2.3. INITIAL SETUP

The instructions in this guide follow on from and assume an OpenShift instance similar to that created in the [OpenShift Primer](#).

2.4. GETTING STARTED

After you have completed the [Section 2.3, “Initial Setup”](#) instructions, this topic helps you get started with the JBoss xPaaS EAP image by performing the required preliminary steps before you can install the image on OpenShift. This process consists of the following steps:

- 🔗 **Step 1:** Create project
- 🔗 **Step 2:** Create service account
- 🔗 **Step 3:** Create keystore from SSL certificate
- 🔗 **Step 4:** Create secret from keystore
- 🔗 **Step 5:** Add secret to service account
- 🔗 **Step 6:** Create and deploy EAP application

The following instructions describe how to perform each step.

Step 1: Create a new project in OpenShift

A project allows a group of users to organize and manage content separately from other groups. Create a project in OpenShift with the following command.

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

You can then make this new project to be the current project with the following command:

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

Step 2: Create an EAP service account in your project

Service accounts are API objects that exist within each project. Create a service account named **eap-service-account** in the OpenShift project that you created in step 1. For the EAP 7 image specify the service account name to be **eap7-service-account**.

```
$ oc create serviceaccount eap-service-account -n <project-name>
```

After creating the service account, configure the access permissions for it with the following command, specifying the correct name depending on the EAP image version.

```
$ oc policy add-role-to-user view system:serviceaccount:$(oc project -q):eap-service-account -n $(oc project -q)
```



Note

The service account that you create must be configured with the correct permissions with the ability to view pods in Kubernetes. This is required in order for clustering with the xPaaS EAP image to work. You can view the top of the log files to see whether the correct service account permissions have been configured.

Step 3: Create a keystore from SSL certificate

The xPaaS EAP image requires a keystore to be imported to properly install and configure the image on your OpenShift instance. Note that self-signed certificates do not provide secure communication and are intended for internal testing purposes.

Warning

For production environments Red Hat recommends that you use your own SSL certificate purchased from a verified Certificate Authority (CA) for SSL-encrypted connections (HTTPS).

See [Generate a SSL Encryption Key and Certificate](#) for more information on how to create a keystore with self-signed or purchased SSL certificates.

Step 4: Create a secret from the keystore

Next, create a secret from the keystore that you created in step 1 with the following command.

```
$ oc secret new <secret-name> <keystore-filename>.jks
```

Step 5: Add the secret to your service account

Now add the secret created in step 3 to the **eap-service-account** that was created in step 2. You can do this with the following command.

```
$ oc secrets add serviceaccount/eap-service-account secret/<secret-name>
```

Step 6: Create and deploy the EAP application

You can now create an EAP application using the defined image, or you can use the basic S2I template.

To create an EAP application using the defined image, run the following command.

```
$ oc new-app <jboss-eap-7/eap70-openshift>
```

Alternatively, you can create an EAP application using the basic S2I template with the following command.

```
$ oc new-app <eap7-basic-s2i>
```

2.5. CONFIGURING THE XPAAS EAP IMAGE

The recommended method to run and configure the xPaaS JBoss EAP image is to use the OpenShift S2I process together with the application template parameters and environment variables.

**Note**

The variable **EAP_HOME** is used to denote the path to the JBoss EAP installation. Replace this variable with the actual path to your JBoss EAP installation.

The S2I process for the xPaaS JBoss EAP image works as follows:

1. If a **pom.xml** file is present in the source repository, a Maven build process is triggered that uses the contents of the **\$MAVEN_ARGS** environment variable. Although you can specify arguments or options with the **\$MAVEN_ARGS** environment variable, Red Hat recommends that you use the **\$MAVEN_ARGS_APPEND** environment variable to do this. The **\$MAVEN_ARGS_APPEND** variable takes the default arguments from **\$MAVEN_ARGS** and appends the options from **\$MAVEN_ARGS_APPEND** to it. By default, the OpenShift profile uses the Maven **package** goal which includes system properties for skipping tests (**-DskipTests**) and enabling the Red Hat GA repository (**-Dcom.redhat.xpaas.repo**). The results of a successful Maven build are copied to **EAP_HOME/standalone/deployments**. This includes all JAR, WAR, and EAR files from the source repository specified by the **\$ARTIFACT_DIR** environment variable. The default value of **\$ARTIFACT_DIR** is the target directory.
2. **EAP_HOME/standalone/deployments** is the artifacts directory, which is specified with the **\$ARTIFACT_DIR** environment variable.
3. All files in the configuration source repository directory are copied to **EAP_HOME/standalone/configuration**. If you want to use a custom JBoss EAP configuration file, it should be named **standalone-openshift.xml**.
4. All files in the modules source repository directory are copied to **EAP_HOME/modules**.

CHAPTER 3. REFERENCE INFORMATION



Note

The content in this section is derived from the engineering documentation for this image. It is provided for reference as it can be useful for development purposes, and for testing beyond the scope of the product documentation

3.1. DATASOURCES

Datasources are automatically created based on the value of some environment variables.

The most important is the **DB_SERVICE_PREFIX_MAPPING** environment variable that defines JNDI mappings for data sources. This variable must be set to a comma-separated list of **<name>-<database_type>=<PREFIX>** triplets, where **name** is used as the pool-name in the data source, **database_type** determines what database driver to use, and **PREFIX** is the prefix used in the names of environment variables, which are used to configure the data source.

3.1.1. JNDI mappings for datasources

For each **<name>-<database_type>=<PREFIX>** triplet in the **DB_SERVICE_PREFIX_MAPPING** environment variable, a separate datasource will be created by the launch script, which is executed when running the image.



Note

The first part (before the equal sign) of the **DB_SERVICE_PREFIX_MAPPING** should be lowercase.

The **<database_type>** will determine the driver for the datasource. Currently, only **postgresql** and **mysql** are supported.

Warning

When choosing a name for the **<name>** parameter, do not use any special characters.

3.1.1.1. Database drivers

Every image contains Java drivers for MySQL, PostgreSQL and MongoDB databases deployed. Datasources are **generated only for MySQL and PostgreSQL databases**.



Note

For MongoDB database there are no JNDI mappings created because this is not a SQL database.

3.1.1.2. Datasource configuration environment variables

Other datasource properties will be configured from the following environment variables:

Variable name	Description	Example value
<NAME>_<DATABASE_TYPE>_SERVICE_HOST	Defines the database server's hostname or IP to be used in the datasource's connection-url property.	192.168.1.3
<NAME>_<DATABASE_TYPE>_SERVICE_PORT	Defines the database server's port for the datasource.	5432
<PREFIX>_JNDI	Defines the JNDI name for the datasource. Defaults to java:jboss/datasources/<name>_<database_type> , where name and database_type are taken from the triplet described above. This setting is useful if you want to override the default generated JNDI name.	java:jboss/datasources/test-postgresql
<PREFIX>_USERNAME	Defines the username for the datasource.	admin
<PREFIX>_PASSWORD	Defines the password for the datasource.	password
<PREFIX>_DATABASE	Defines the database name for the datasource.	myDatabase
<PREFIX>_TX_ISOLATION	Defines the java.sql.Connection transaction isolation level for the datasource.	TRANSACTION_READ_UNCOMMITTED
<PREFIX>_TX_MIN_POOL_SIZE	Defines the minimum pool size option for the datasource.	1

Variable name	Description	Example value
<PREFIX>_TX_MAX_POOL_SIZE	Defines the maximum pool size option for the datasource.	20

When running this image in OpenShift, the **<NAME>_<DATABASE_TYPE>_SERVICE_HOST** and **<NAME>_<DATABASE_TYPE>_SERVICE_PORT** environment variables are set up automatically from the database service definition in the OpenShift application template, while the others are configured in the template directly (as **env** entries in container definitions under each pod template).

3.1.1.3. Examples

These examples show how value of the **DB_SERVICE_PREFIX_MAPPING** environment variable influences datasource creation.

3.1.1.3.1. Single mapping

Consider value **test-postgresql=TEST**.

This will create a datasource with **java:jboss/datasources/test_postgresql** name. Additionally all the required settings like password and username will be expected to be provided as env variables with the **TEST_** prefix, for example **TEST_USERNAME** and **TEST_PASSWORD**.

3.1.1.3.2. Multiple mappings

You can also specify multiple database mappings. Consider following value for the **DB_SERVICE_PREFIX_MAPPING** environment variable: **cloud-postgresql=CLOUD, test-mysql=TEST_MYSQL**.



Note

Multiple datasource mappings should be separated with comma.

This will create two datasources:

1. **java:jboss/datasources/test_mysql**, and
2. **java:jboss/datasources/cloud_postgresql**.

MySQL datasource configuration (username, etc) will be expected with the **TEST_MYSQL** prefix, for example **TEST_MYSQL_USERNAME**, whereas for the PostgreSQL datasource it'll expect beexpected with the **CLOUD_** prefix, for example **CLOUD_USERNAME**.

3.2. CLUSTERING

Clustering is achieved through one of two discovery mechanisms: Kubernetes or DNS. This is done by configuring the JGroups protocol stack in standalone-openshift.xml with either the `<openshift.KUBE_PING/>` or `<openshift.DNS_PING/>` elements. Out of the box, **KUBE_PING** is the pre-configured and supported protocol.

For **KUBE_PING** to work, however, the following steps must be taken:

1. The **OPENSIFT_KUBE_PING_NAMESPACE** environment variable must be set. If not set, the server will behave as a single-node cluster (a "cluster of one").
For example:

```
OPENSIFT_KUBE_PING_NAMESPACE=<project_name>
```

2. The **OPENSIFT_KUBE_PING_LABELS** environment variables should be set. If not set, pods outside of your application (albeit in your namespace) will try to join.
For example:

```
OPENSIFT_KUBE_PING_LABELS=application=<app_name>
```

3. Authorization must be granted to the service account the pod is running under to be allowed to access Kubernetes' REST api. This is done on the command line.
For example:
Using the default service account in the myproject namespace:

```
oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default -n $(oc project -q)
```

Using the eap-service-account in the project namespace:

```
oc policy add-role-to-user view system:serviceaccount:$(oc project -q):eap-service-account -n $(oc project -q)
```



Note

See [Section 2.4, "Getting Started"](#) for more information on adding policies to service accounts.

3.3. SECURITY DOMAINS

To configure a new Security Domain, the user must define the **SECDOMAIN_NAME** environment variable.

This will result in the creation of a security domain named after the environment variable. The user may also define the following environment variables to customize the domain:

Variable name	Description	Example value
---------------	-------------	---------------

Variable name	Description	Example value
SECDOMAIN_NAME	Define in order to enable the definition of an additional security domain.	myDomain
SECDOMAIN_PASSWORD_STACKING	If defined, the password-stacking module option is enabled and set to the value <code>useFirstPass</code> .	true
SECDOMAIN_LOGIN_MODULE	The login module to be used. Defaults to UsersRoles	UsersRoles
SECDOMAIN_USERS_PROPERTIES	The name of the properties file containing user definitions. Defaults to users.properties	users.properties
SECDOMAIN_ROLES_PROPERTIES	The name of the properties file containing role definitions. Defaults to roles.properties	roles.properties

3.4. HTTPS

3.4.1. Environment variables

Variable name	Description	Example value
HTTPS_NAME	If defined along with HTTPS_PASSWORD and HTTPS_KEYSTORE , enable HTTPS and set the SSL name.	example.com
HTTPS_PASSWORD	If defined along with HTTPS_NAME and HTTPS_KEYSTORE , enable HTTPS and set the SSL key password.	passw0rd

Variable name	Description	Example value
HTTPS_KEYSTORE	If defined along with HTTPS_PASSWORD and HTTPS_NAME , enable HTTPS and set the SSL certificate key file to a relative path under \$JBOSS_HOME/standalone/configuration	ssl.key

3.5. ADMINISTRATION

3.5.1. Environment Variables

Variable name	Description	Example value
ADMIN_USERNAME	If both this and ADMIN_PASSWORD are defined, used for the EAP management port user name.	eapadmin
ADMIN_PASSWORD	If defined, an admin user is defined for accessing the management port, with this value as password.	password

3.6. S2I

The image includes S2I scripts and maven.

Maven is currently only supported as a build tool for applications that are supposed to be deployed on JBoss EAP-based containers (or related/descendant images) on OpenShift.

Currently WAR, EAR, and JAR deployments are supported.

3.6.1. Custom configuration

It is possible to add custom configuration files for the image. All files put into `configuration/` directory will be copied into **\$JBOSS_HOME/standalone/configuration/**. For example to override the default configuration used in the image, just add a custom `standalone-openshift.xml` into the `configuration/` directory.

3.6.1.1. Custom modules

It is possible to add custom modules. All files from the `modules/` directory will be copied into `$JBOSS_HOME/modules/`.

3.6.2. Deployment Artifacts

By default, artifacts from the source **target** directory will be deployed. To deploy from different directories set the `ARTIFACT_DIR` environment variable in the BuildConfig definition.

`ARTIFACT_DIR` is a comma-delimited list. For example:

`ARTIFACT_DIR=app1/target,app2/target,app3/target`

3.6.3. Scripts

run

this script uses the **launch.sh** script that configures and starts EAP with the **standalone-openshift.xml** configuration.

assemble

uses Maven to build the source, create a package (war) and move it to the `$JBOSS_HOME/standalone/deployments` directory.

3.6.4. Environment variables

You can influence the way the build is executed by supplying environment variables to the **s2i build** command. The environment variables that can be supplied are:

Variable name	Description	Example value
ARTIFACT_DIR	.war , .ear , and .jar files from this directory will be copied into the deployments directory.	target
HTTP_PROXY_HOST	Hostname or IP address of a HTTP proxy for Maven to use.	192.168.1.1
HTTP_PROXY_PORT	TCP Port of a HTTP proxy for Maven to use.	8080
HTTP_PROXY_USERNAME	If supplied with HTTP_PROXY_PASSWORD , use credentials for HTTP proxy.	myusername

Variable name	Description	Example value
HTTP_PROXY_PASSWORD	If supplied with HTTP_PROXY_USERNAME , use credentials for HTTP proxy.	mypassword
HTTP_PROXY_NONPROXYHOSTS	If supplied, a configured HTTP proxy will ignore these hosts.	some.example.org *.example.net
MAVEN_ARGS	Overrides the arguments supplied to maven during build.	-e -Popenshift -DskipTests -Dcom.redhat.xpaas.repo.redhatga package
MAVEN_ARGS_APPEND	Appends user arguments supplied to maven during build.	-Dfoo=bar
MAVEN_MIRROR_URL	URL of a Maven Mirror/repository manager to configure.	http://10.0.0.1:8080/repository/internal/
MAVEN_CLEAR_REPO	Optionally clear the local maven repository after the build.	true
APP_DATADIR	If defined, directory in the source from where data files are copied.	mydata
DATA_DIR	Directory in the image where data from \$APP_DATADIR will be copied.	\$JBOSS_HOME/data

3.7. SSO

This image contains support for Red Hat SSO-enabled applications.



Note

See [Red Hat xPaaS SSO Image](#) documentation for more information on how to deploy the xPaaS SSO image with the EAP image.

3.7.1. Environment variables

Variable name	Description	Example value	Default value
SSO_URI	URI of the SSO server	-	-
SSO_REALM	SSO realm for the deployed application(s)	-	-
SSO_PUBLIC_KEY	Public key of the SSO Realm. This field is optional but if omitted can leave the applications vulnerable to man-in-middle attacks	-	-
SSO_USERNAME	SSO User required to access the SSO REST API	mySsoUser	-
SSO_PASSWORD	Password for SSO_USERNAME	6fedmL3P	-
SSO_SAML_KEYSTORE	Keystore location for SAML	-	/etc/sso-saml-secret-volume/keystore.jks
SSO_SAML_KEYSTORE_PASSWORD	Keystore password for SAML	-	mykeystorepass
SSO_SAML_CERTIFICATE_NAME	Alias for keys/certificate to use for SAML	-	jboss
SSO_BEARER_ONLY	Optional. SSO Client Access Type	true	-

Variable name	Description	Example value	Default value
SSO_CLIENT	Path for SSO redirects back to the application	Defaults to match module-name	-
SSO_ENABLE_CORS	Optionally enable CORS for SSO applications	true	-
SSO_SECRET	The SSO Client Secret for Confidential Access	KZ1Qylq4	-
SSO_SECURE_SSL_CONNECTIONS	If true SSL communication between EAP and the SSO Server will be secure (i.e. certificate validation is enabled with curl)	false	-