DigiTalk

Jboss: Migrating Applications

From JBoss EAP 7.x to JBoss EAP 8.0

A Comprehensive Technical Guide

Based on the Red Hat Developer article

14.1.2 & 15.1.1
DT246055-03
July 2026

Jboss Server Administration
DT246055-03

July, 2026

DigiTalk

Table of Contents

1. Introduction

Red Hat JBoss Enterprise Application Platform (JBoss EAP) 8.0 is a major release that brings support for Jakarta EE 10, a significant departure from the Jakarta EE 8 specification supported by JBoss EAP 7. This document explains the key architectural and feature changes introduced in JBoss EAP 8.0 and provides a detailed, step-by-step walkthrough for migrating an application and its server configuration from JBoss EAP 7.4 to JBoss EAP 8.0.

The guide draws on a practical example — a modified version of the JBoss EAP 7.4 "kitchensink" quickstart connected to a MySQL database — to demonstrate both the code-level changes (driven by the javax → jakarta namespace migration) and the server configuration changes (driven by the move to the Elytron security subsystem and a new provisioning model).

Two Red Hat tools are central to this process:

- **Migration Toolkit for Applications (MTA) / Migration Toolkit for Runtimes (MTR) —** analyzes application source code and binaries and produces a report of the code changes required to migrate to the target platform.
- **JBoss Server Migration Tool —** automates the migration of server configuration files (e.g., standalone.xml), including modules, drivers, datasources, and legacy security configuration.

2. Key Changes in JBoss EAP 8.0

JBoss EAP 8.0 introduces support for Jakarta EE 10 and a number of platform-level changes relative to JBoss EAP 7.x. The most significant of these are summarized below.

- **Removal of PicketBox —** Deprecated since JBoss EAP 6.4. Legacy security configurations must be migrated to the Elytron security subsystem. Applications already using Elytron require no changes.
- **Removal of PicketLink —** Also deprecated since JBoss EAP 6.4. Red Hat Single Sign-On (RH-SSO) is recommended as the replacement for single sign-on functionality.
- **OpenID Connect (OIDC) support —** Now provided by the elytron-oidc-client subsystem rather than the separate RH-SSO OIDC adapter.
- **SAML support —** Provided via the RH-SSO SAML Galleon Feature Pack.
- **Hibernate 6.x —** Applications must be migrated from Hibernate Search 5.11.x to 6.0.x; see the Hibernate Search migration guide for details.
- **Removal of JDK 8 support —** JDK 11 or JDK 17 is now required to run JBoss EAP 8.0.
- **Jolokia and Prometheus removed —** These are no longer supported by Red Hat. Metrics are instead exposed through the server metrics endpoint at <server address>:<management port>/metrics.
- **Apache Log4j version 1 APIs removed —** Applications that do not bundle their own log4j.jar and Log4j configuration must be updated accordingly.
- **New eap-maven-plugin —** Built on Galleon technology, this plugin assembles a server based only on the layers an application actually needs. This reduces server size, memory footprint, and attack surface — if a JAR affected by a CVE is not present, the server is not vulnerable to that CVE. It also supports execution of JBoss EAP CLI scripts to customize server configuration. OpenShift Source-to-Image (S2I) builds have been redesigned around this plugin, and the OpenShift images for JBoss EAP 8 no longer bundle EAP runtime binaries as they did for EAP 6 and EAP 7.
- **New provisioning system —** Provides greater consistency across bare metal, virtual machine, and cloud deployment targets, and reduces disparity between test/staging and production environments.

3. Migration Overview

Migrating an application from JBoss EAP 7.4 to JBoss EAP 8.0 requires code changes because of the move from Jakarta EE 8 to Jakarta EE 10 — most notably, converting the javax.* namespace to jakarta.*. These changes are documented in Red Hat's Jakarta EE 8 to EE 10 migration article.

This guide focuses on two areas:

- The code changes required, as identified by the Migration Toolkit for Runtimes (MTR).
- The configuration changes needed for the standalone.xml server configuration file, performed using the JBoss Server Migration Tool.

The example application used throughout is based on the JBoss EAP 7.4 kitchensink quickstart, modified to connect to a MySQL database. This allows the migration of custom JBoss modules and JDBC drivers to be demonstrated in addition to the code and configuration changes.

4. Prerequisites: Setting Up JBoss EAP 7.4

Before performing the migration, set up a working instance of JBoss EAP 7.4 running the sample application. This establishes a baseline to migrate from.

4.1 Start a MySQL Database

Use Podman or Docker to run a MySQL instance with the correct configuration:

```
podman run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=eap mysql
```

4.2 Download and Extract JBoss EAP 7.4

Download the JBoss EAP 7.4 zip distribution (a Red Hat account is required) and extract it to a local folder, for example ~/jboss-eap-74.

Set the EAP_HOME environment variable:

```
export JBOSS_HOME=~/jboss-eap-74
```

Start JBoss EAP 7.4:

```
$JBOSS_HOME/bin/standalone.sh
```

In a new terminal window, set $JBOSS_HOME again:

```
export JBOSS_HOME=~/jboss-eap-74
```

4.3 Get the Example Application

Clone the example application repository and change into the kitchensink project directory:

```
git clone https://github.com/deewhyweb/eap-quickstarts.git
cd eap-quickstarts/kitchensink
```

4.4 Copy Custom Modules

Copy the contents of the project's modules folder into the JBoss EAP 7.4 modules directory:

```
cp -r modules/* $JBOSS_HOME/modules
```

4.5 Install the MySQL JDBC Driver

Download the MySQL Connector/J driver and copy it into the appropriate module directory:

```
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j-8.0.33.zip
unzip mysql-connector-j-8.0.33.zip
cp mysql-connector-j-8.0.33/mysql-connector-j-8.0.33.jar $JBOSS_HOME/modules/com/mysql/main/
```

4.6 Configure the Datasource via the JBoss CLI

Connect to the running server using the JBoss EAP CLI:

```
$JBOSS_HOME/bin/jboss-cli.sh --connect
```

From the CLI, register the MySQL JDBC driver and create the datasource:

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql)

data-source add --name=mysql --jndi-name=java:/jdbc/mysql --driver-name=mysql \
  --connection-url=jdbc:mysql://127.0.0.1:3306/eap --user-name=root --password=root
```

4.7 Build and Deploy the Application

Build and deploy the application to JBoss EAP 7.4 with Maven:

```
mvn clean install wildfly:deploy
```

Once deployed, the application should be reachable at http://127.0.0.1:8080/kitchensink.

Note: Once the application is confirmed to be running correctly on JBoss EAP 7.4, shut down the server before starting the migration. Create an environment variable pointing at the EAP 7.4 installation: export EAP_PREVIOUS_HOME=~/jboss-eap-74

5. Installing JBoss EAP 8.0

Download the JBoss EAP 8.0 installation tool and extract it to a local folder, for example ~/jboss-eap-8-installer. Then create a target directory for the new server:

```
mkdir ~/jboss-eap-8
```

Run the JBoss EAP 8 provisioning tool to install the server:

```
~/jboss-eap-8-installer/bin/jboss-eap-installation-manager.sh install \
  --profile=eap-8.0 --dir=$HOME/jboss-eap-8
```

Follow the on-screen instructions to complete installation, then set EAP_HOME to point to the new server:

```
export JBOSS_HOME=~/jboss-eap-8
```

6. Server Configuration Migration

The sample application required custom modules and a JDBC driver to be added to the JBoss EAP 7.4 configuration. These must be carried over correctly to JBoss EAP 8.0. Rather than performing this manually, the JBoss EAP Server Migration Tool automates the process.

Download the server migration tool, extract the archive, and change into the extracted directory. Then run:

```
./jboss-server-migration.sh -s $EAP_PREVIOUS_HOME -t $JBOSS_HOME
```

6.1 Interactive Migration Session

The tool runs interactively, asking a series of yes/no questions. The key prompts and recommended answers for this scenario are as follows:

```
------------------------------------------------------------
----  JBoss Server Migration Tool  ----------------------
------------------------------------------------------------

Retrieving servers...
 INFO  SOURCE server name: JBoss EAP, version: 7.4.0.GA.
 INFO  TARGET server name: JBoss EAP, version: 8.0.0.GA.

Server migration starting...
 INFO  --- Migrating modules requested by environment...
 INFO  No modules to migrate.

Migrate the source's standalone server?  yes/no?
```

Answer yes to migrate the standalone configuration.

```
 INFO  --- Migrating standalone server...
 INFO  Source's standalone content migrated.
 INFO  Source's standalone configurations found:
       [standalone-full-ha.xml, standalone-full.xml, standalone-ha.xml,
        standalone-load-balancer.xml, standalone.xml]

Migrate all configurations?  yes/no?
```

Answer no — select configurations individually. Answer no for standalone-full-ha.xml, standalone-full.xml, standalone-ha.xml, and standalone-load-balancer.xml, then yes for standalone.xml, since that is the configuration in use for this example.

```
 INFO  Migrating standalone configuration standalone.xml...
 WARN  Migration of legacy security domain jboss-web-policy's
       authorization is not supported and will be ignored.
 WARN  Migration of legacy security domain jaspitest's
       authentication-jaspi is not supported and will be ignored.
 WARN  Migration of legacy security domain jboss-ejb-policy's
       authorization is not supported and will be ignored.
```

```
 INFO  Legacy security XML configuration retrieved.
 INFO  Unsupported extensions removed: [org.jboss.as.security]
 INFO  Unsupported subsystems removed: [urn:jboss:domain:security:2.0]
 INFO  Referenced paths migrated.
 INFO  Legacy security realms removed from XML configuration.
 INFO  Legacy security realm ManagementRealm migrated to Elytron.
 INFO  Legacy security realm ApplicationRealm migrated to Elytron.
 INFO  Legacy security domain other migrated to Elytron.

Migrate the source's managed domain?  yes/no?
```

Answer no, since this example does not use a managed domain. The tool then completes and prints a task summary:

```
--------------------------------------------------------------------------
 Task Summary
--------------------------------------------------------------------------
 server ..................................................  SUCCESS
  standalone .............................................  SUCCESS
   contents.standalone.migrate-content-dir ...............  SUCCESS
   standalone-configurations .............................  SUCCESS
    standalone-configuration(source=standalone.xml) .....  SUCCESS
--------------------------------------------------------------------------
 Migration Result: SUCCESS
--------------------------------------------------------------------------
```

Note: During migration, legacy PicketBox security realms and domains (ManagementRealm, ApplicationRealm, and the “other” security domain) are automatically migrated to the Elytron subsystem. Any legacy security domain configuration that Elytron does not support (such as certain authorization or JASPI-based authentication configurations) will be flagged and ignored, and must be reviewed and re-implemented manually.

6.2 Starting the Migrated Server

Once the migration completes, start the new JBoss EAP 8.0 server:

```
$JBOSS_HOME/bin/standalone.sh
```

7. Verifying the Server Migration

With the server running, connect using the JBoss CLI and confirm the MySQL driver was migrated correctly:

```
$JBOSS_HOME/bin/jboss-cli.sh --connect
/subsystem=datasources:installed-drivers-list
```

The expected output confirms that the MySQL driver is registered:

```
{
    "outcome" => "success",
    "result" => [
        {
            "driver-name" => "mysql",
            "deployment-name" => undefined,
```

```
            "driver-module-name" => "com.mysql",
            "module-slot" => "main",
            "driver-class-name" => "com.mysql.cj.jdbc.Driver",
            "driver-major-version" => 8,
            "driver-minor-version" => 0,
            "jdbc-compliant" => false
        }
    ]
}
```

Next, verify the datasource connection itself:

```
/subsystem=datasources/data-source=mysql:test-connection-in-pool
```

A successful connection returns:

```
{
    "outcome" => "success",
    "result" => [true]
}
```

At this point, the required modules, drivers, and datasources are confirmed present and working correctly on JBoss EAP 8.0, and the migration can proceed to code analysis.

8. Code Analysis with the Migration Toolkit for Runtimes

The Migration Toolkit for Runtimes (MTR) analyzes application source or binaries and reports the code changes needed for the target platform. Download the latest MTR release, then launch it:

```
./run_windup.sh
```

Navigate to the MTR web UI at http://127.0.0.1:8080/windup-ui.

8.1 Create a Project and Upload the Application

- From the MTR landing page, click Create Project.
- Enter a project name (e.g., eap7-eap8) and click Next.
- Upload the application artifact to analyze — in this case, the kitchensink.war build located in eap-quickstarts/kitchensink/target — and click Next.

8.2 Select the Transformation Target

- On the target selection screen, choose “eap8” from the “Application server migration to EAP 7” box, then click Next.
- On the packages screen, add org to the list of selected packages and click Next.
- Click through the remaining pages, then select Save and Run to start the analysis.

8.3 Review the Analysis Report

When the analysis completes, click the graph icon next to the analysis row to load the report. For the kitchensink sample application, the tool estimates 87 story points to complete the migration.

Note: Story points here are a relative sizing metric, not a time estimate in hours or days — they are most useful for comparing the relative complexity of different applications' migrations.

Click kitchensink.war to view the detailed list of issues, then open the Issues tab and filter to Migration Mandatory items — the changes that must be made for the application to run on JBoss EAP 8.0.

The reported issues align with expectations: the bulk of the work involves converting javax package references to jakarta, along with a small number of other changes, such as updating the Jakarta Server Faces (JSF) version used by the application to match the version bundled with JBoss EAP 8.0.

Two approaches are available for applying these changes:

- Go through each reported issue individually and apply the code changes manually.
- Use an automated refactoring tool such as OpenRewrite to apply the javax → jakarta namespace migration in bulk (see Red Hat's guide on using OpenRewrite for Java EE 8 to Jakarta EE 9 migration).

9. Applying Code Changes: pom.xml Example

One of the mandatory changes reported by MTR is an update to the project's pom.xml dependency declarations — specifically the groupId, artifactId, and the version.server.bom property, which must be updated to reference JBoss EAP 8.0.

Replace the following line in pom.xml:

```
<version.server.bom>7.4.0.GA</version.server.bom>
```

with:

```
<version.server.bom>8.0.0.GA-redhat-00009</version.server.bom>
```

Similar mandatory changes reported elsewhere in the project (primarily javax.* → jakarta.* package renames) should be applied throughout the codebase before proceeding.

10. Building and Deploying to JBoss EAP 8.0

With all reported mandatory code changes applied and the JBoss EAP 8.0 server running, build and deploy the application using Maven:

```
mvn clean install wildfly:deploy
```

Once the Maven build completes successfully, the application should be available at the same address as before:

```
http://127.0.0.1:8080/kitchensink
```

At this point, both the application code and the server configuration have been fully migrated from JBoss EAP 7.4 to JBoss EAP 8.0.

11. Conclusion

JBoss EAP 8.0 represents a substantial platform upgrade, bringing support for Jakarta EE 10, the removal of legacy security subsystems (PicketBox and PicketLink) in favor of Elytron, a new Galleon-based provisioning model, and updated minimum JDK requirements.

Migrating an existing JBoss EAP 7.4 application to JBoss EAP 8.0 involves two parallel workstreams:

- **Server configuration migration —** automated using the JBoss Server Migration Tool, which migrates modules, drivers, datasources, and legacy security realms/domains into the Elytron subsystem.
- **Application code migration —** guided by the Migration Toolkit for Runtimes (MTR), which identifies the code changes required — chiefly the javax → jakarta namespace conversion — and can be paired with automated refactoring tools such as OpenRewrite to accelerate the work.

Following the process demonstrated in this document — setting up a baseline JBoss EAP 7.4 environment, provisioning JBoss EAP 8.0, running the server migration tool, analyzing the application with MTR, and applying the reported code changes — provides a reliable, repeatable path for moving production applications onto JBoss EAP 8.0.