How to migrate apps from JBoss EAP 7.x to JBoss EAP 8.0

JBoss EAP 8.0 provides support for Jakarta EE 10. Jakarta EE 10 brings a huge change to Jakarta EE compared to the Jakarta EE 8 specifications supported by EAP 7. JBoss EAP 8 also introduces changes in the following areas:

- **Removal of Picketbox:** This has been deprecated since the release of JBoss EAP 6.4. Any legacy security configurations must be migrated to the Elytron security subsystem. EAP users who have already migrated to Elytron don't need to make any changes.
- **Removal of PicketLink**: This has been deprecated since the release of JBoss EAP 6.4. Leveraging RH SSO is recommended for the single sign-on functionality previously offered with PicketLink.
- **Support for OpenID Connect (OIDC)** will now be provided by the elytron-oidc-client subsystem instead of the separate RH-SSO OIDC adapter.
- **SAML support** is provided by the RH-SSO SAML Galleon Feature Pack.
- **Removal of JDK 8:** JDK 11 or JDK 17 is now required.
- **Jolokia and Prometheus removed:** Red Hat will no longer support these features. JBoss EAP server exposes metrics through the server metrics endpoint: <server address>:<management port>/metrics.
- **Apache Log4j version 1 APIs removed:** If your applications do not package log4j.jar and Log4j configuration as part of the application, then you must update them.
- **New maven plugin:** The new eap-maven-plugin uses Galleon technology to build the server based on the layer configuration needed in your application. This results in a reduced size and memory footprint, plus a reduced attack surface. If the JAR impacted by a CVE is not present, EAP will not be vulnerable to the CVE. The

JBoss EAP 8 OpenShift images don't contain EAP runtime binaries like it used to in EAP 6 and EAP 7. This plugin also supports the execution of JBoss EAP CLI script files to customize your server configuration. OpenShift Source to Image builds (S2I) incorporate this change and have been redesigned to leverage the eap-maven-plugin to create the server.

- **New provisioning system:** JBoss EAP 8.0 introduces a new provisioning system providing increased consistency across all deployment targets (bare metal, VM, or cloud) and reducing disparity between testing/staging and production environments.

Migrating from JBoss EAP 7.4 to JBoss EAP 8.0

Moving applications from JBoss EAP 7.4 to JBoss EAP 8.0 will require code changes (due to the move from Jakarta EE 8 to Jakarta EE 10 (e.g., converting from the "javax" namespace to "jakarta" namespace).

Prerequisites

Before starting, we will set up an instance of JBoss EAP 7.4 with a running application. This will help demonstrate the server configuration migration process.

We will use the JBoss EAP 7.4 quickstarts kitchensink application with some modifications as follows:

- The application has been updated to connect to a MySQL database. This will allow us to test the migration of J2EE modules and drivers.
- Java Server Faces has been updated from version 2.2 to 2.3, as described in the KCS article.

We can use Podman or Docker to run an instance of MySQL with the correct configuration with the following command:

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

Download the JBoss EAP 7.4 zip distribution. Extract the distribution into a local folder (e.g., `~/jboss-eap-74`). If you don't already have a Red Hat account, you need to create one before you can log in to download this archive.

Set the `EAP_HOME` environment variable:

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

Now we can start JBoss EAP 7.4 with the following command:

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

Open a new terminal window and set `$JBOSS_HOME`:

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

Check out this example application.

Change the directory into the `eap-quickstarts/kitchensink` folder.

Copy the contents of the `modules` folder to the `~/jboss-eap-74/modules` folder.

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

Download the MySQL JDBC driver by running the following commands:

```
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/
```

Run the JBoss EAP CLI:

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

From the JBoss CLI, enter the following commands to configure the MySQL driver and data source:

```
/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
```

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

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

Once this application is deployed successfully, we should be able to access it here: http://127.0.0.1:8080/kitchensink.

We now have a running instance of our application running in JBoss EAP 7.4 ready to migrate to JBoss EAP 8.0. Before starting the migration, shut down the JBoss EAP 7.4 server and create an environment variable pointing to the JBoss EAP 7.4 deployment.

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

Download the JBoss EAP 8.0 installation tool here.

Extract the tool into a local folder (e.g., `~/jboss-eap-8-installer`).

Create a folder to deploy JBoss EAP 8.0 (e.g., `~/jboss-eap-8`)

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

Run the JBoss EAP 8 provisioning tool

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

Follow the instructions to install an instance of JBoss EAP 8.0

Set the `EAP_HOME` environment variable as follows:

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

Now we can move on to the migration from JBoss EAP 7.4 to JBoss EAP 8.0..

Server configuration changes

When we deployed our application to JBoss EAP 7.4, we made some changes to the server configuration to add a module and driver to connect to MySQL. We must ensure these modules and drivers are in place and working correctly in our JBoss EAP 8.0 server. We can use the JBoss EAP server migration tool to perform this migration for us.

Download the server migration tool from this link.

Extract the archive to a local folder and change into the new directory. Run the following command:

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

The server migration tool will ask a series of questions during the migration process:

```
-------------------------------------------------------------

----  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?
```

Choose **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?
```

Choose **no**. We want to select the configurations to migrate.

```
Migrate configuration standalone-full-ha.xml ?

yes/no?
```

Choose **no**. We don’t want to migrate standalone-full-ha.xml.

```
Migrate configuration standalone-full.xml ?

yes/no?
```

Choose **no**. We don’t want to migrate standalone-full.xml.

```
Migrate configuration standalone-ha.xml ?

yes/no?
```

Choose **no**. We don’t want to migrate standalone-ha.xml.

```
Migrate configuration standalone-load-balancer.xml ?

yes/no?
```

Choose **no**. We don’t want to migrate standalone-load-balancer.xml.

```
Migrate configuration standalone.xml ?

yes/no?
```

Choose **yes**. We want to migrate standalone.xml.

```
 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.

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by
org.wildfly.extension.elytron.SSLDefinitions
(jar:file:/home/philip/Downloads/wildfly-ee-dist-8.0.0-redhat-
00002/jboss-eap-
```

```
8.0/modules/system/layers/base/org/wildfly/extension/elytron/m
ain/wildfly-elytron-integration-jakarta-19.0.0.Beta16-redhat-
00004.jar!/) to method
com.sun.net.ssl.internal.ssl.Provider.isFIPS()

WARNING: Please consider reporting this to the maintainers of
org.wildfly.extension.elytron.SSLDefinitions

WARNING: Use --illegal-access=warn to enable warnings of
further illegal reflective access operations

WARNING: All illegal access operations will be denied in a
future release

 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?
```

Choose **no**. We are not using a managed domain.

```
Server migration done.


 INFO

-------------------------------------------------------------
---------------------------------------------

 Task Summary

-------------------------------------------------------------
---------------------------------------------



server ..........................................................
.............................. SUCCESS


standalone ......................................................
.............................. SUCCESS
```

```
   contents.standalone.migrate-content-
dir ..................................................
SUCCESS

    contents.standalone.migrate-
content(path=7d/975dfd92d3b7a35f48054ad2190bdf7ebbeb3b/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=b5/fe70630a477a851f755cc04eb5cc3030bb02db/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=92/2cdbb0d087989ad5e6ecbd3c0123a34def19b7/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=14/99d19995fc5835cb1c07eed113447b4fb46c4e/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=3d/25c8107fcc4cf82f234dcf6ad5b8e926e7ec18/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=45/9834b7318bd81b320829432fdef5a77bf93a19/content
) .. SUCCESS

    contents.standalone.migrate-
content(path=38/c53e35a6ab9d8a40016a5429f7115ff94e3ce7/content
) .. SUCCESS

   standalone-
configurations .....................................
..................... SUCCESS

    standalone-
configuration(source=standalone.xml) .......................
................... SUCCESS


------------------------------------------------------------
--------------------------------------------

 Migration Result: SUCCESS

------------------------------------------------------------
--------------------------------------------
```

Once this operation completes, we can start our JBoss EAP 8.0 server with the following command:

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

When the server has started successfully, we can test our configuration with the following commands:

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

/subsystem=datasources:installed-drivers-list
```

The output should show the MySQL driver as follows:

```
{
    "outcome" => "success",
    "result" => [
                {
            "driver-name" => "mysql",
            "deployment-name" => undefined,
            "driver-module-name" => "com.mysql",
            "module-slot" => "main",
            "driver-datasource-class-name" => "",
            "driver-xa-datasource-class-name" => "",
            "datasource-class-info" => undefined,
            "driver-class-name" => "com.mysql.cj.jdbc.Driver",
            "driver-major-version" => 8,
            "driver-minor-version" => 0,
            "jdbc-compliant" => false
        }
```

We can also test our datasource connection with the following JBoss CLI command:

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

A successful connection should result in the following response:

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

We can now be confident our application will have the required drivers and data sources present, so we can move on to the code analysis section.

Code analysis

To perform code analysis of our JBoss EAP 7.4 application, we will use the latest version of the Migration Toolkit for Runtimes. We can download this tool from here.

We launch the MTR tool with the following command: `./run_windup.sh`, then navigate to: `http://127.0.0.1:8080/windup-ui`

From the MTR landing page (Figure 1), click on **Create Project**.

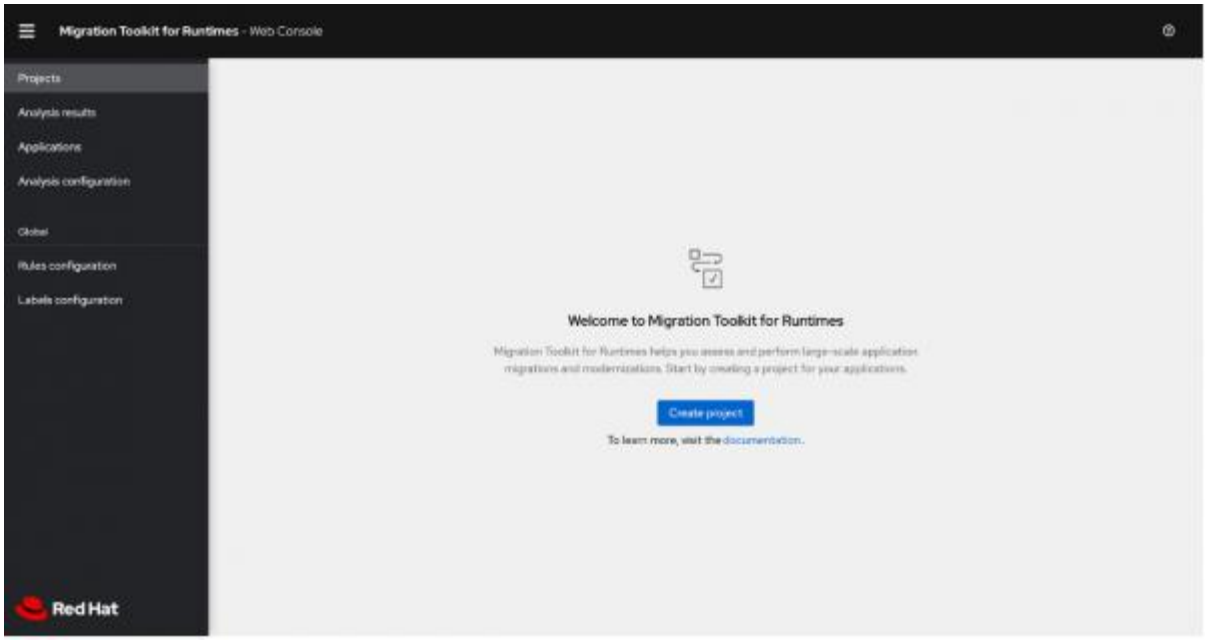

Figure 1: Migration Toolkit for applications start page.

Enter a project name (e.g., eap7-eap8) and click on **Next**.

Then, we need to upload an artifact to analyze (Figure 2).

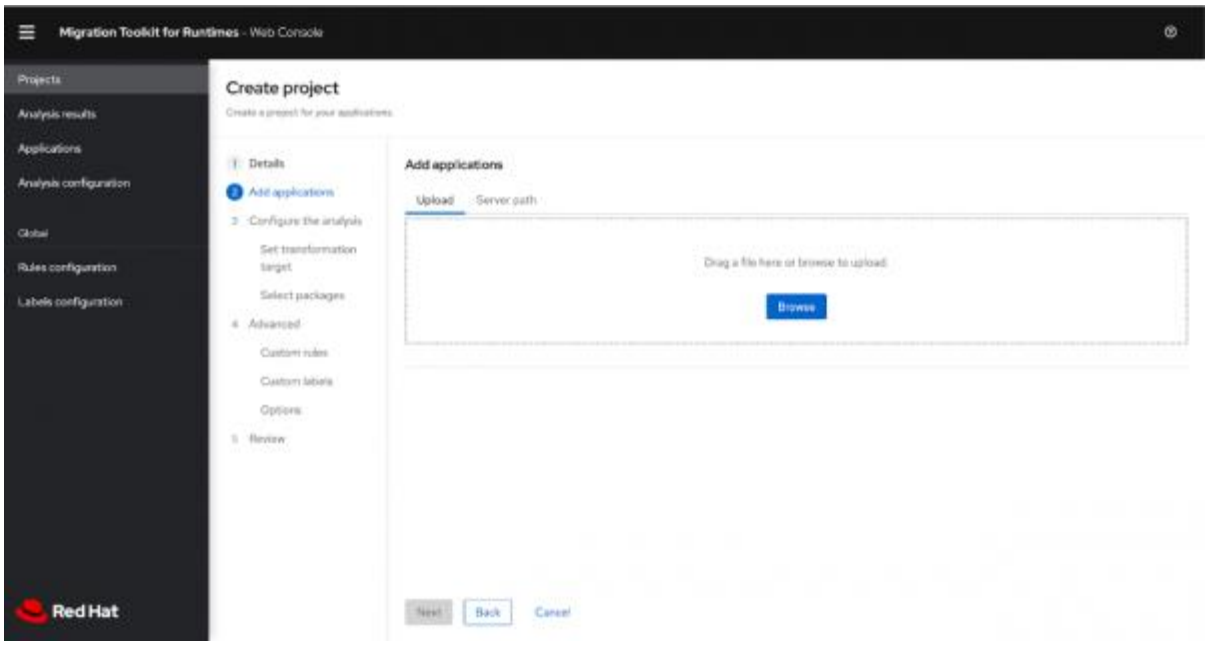

Figure 2: Migration Toolkit for Applications add artifact.

We locate a build of our application in the form of a WAR file and upload it via the web console. The WAR file will be in the `eap-quickstarts/kitchensink/target` folder.

Click on **Next** to progress to the next stage.

Now select the transformation target (Figure 3): select “eap8” from the “Application server migration to EAP 7” box.

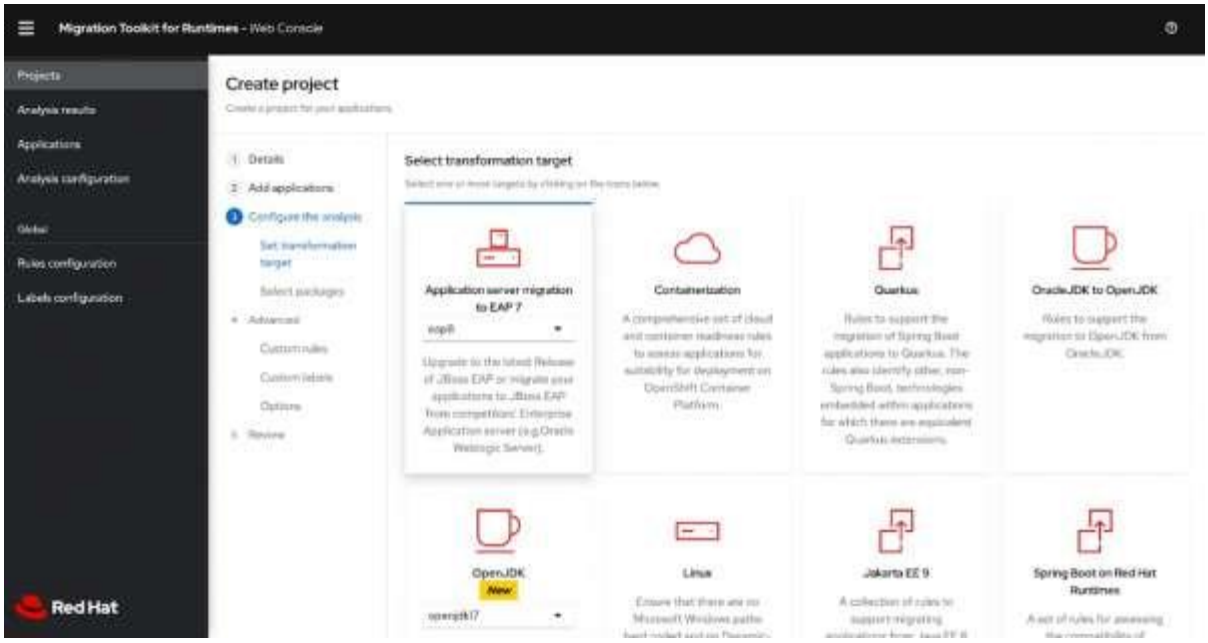

Figure 3: Migration Toolkit for Applications select target page.

Click **Next** to continue.

From the next screen, add **org** to the list of selected packages and click on **Next.**

Click through the pages that follow and then select **Save and Run** to start the analysis. When the analysis finishes, click on the graph symbol on the right of the analysis list row to load the report.

You will see from the report the tool is estimating 87 story points to migrate this application. The estimation of story points does not relate to time (e.g., hours or days). It's more of an indication to measure against reports from other analyses. Click on **kitchensink.war** to view the details of the issues to be addressed (Figure 4).

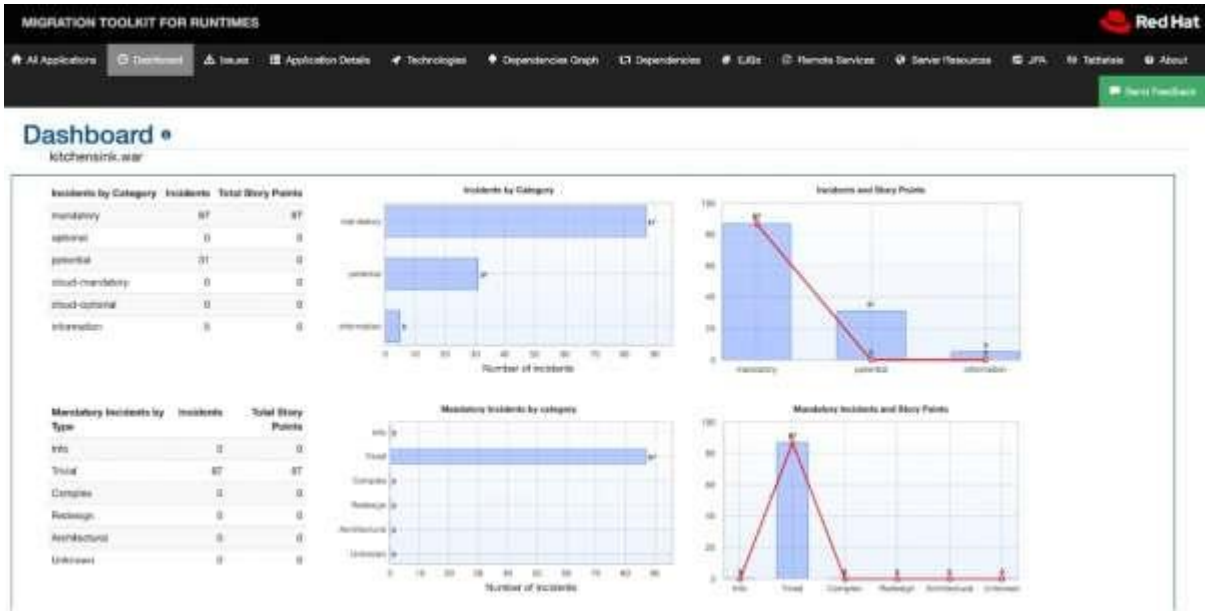

Figure 4: Migration Toolkit for Applications dashboard view.

From the dashboard view, click on the **Issues** tab to view the list of incidents to resolve.

For this article, we will focus on the **Migration Mandatory** items (Figure 5).

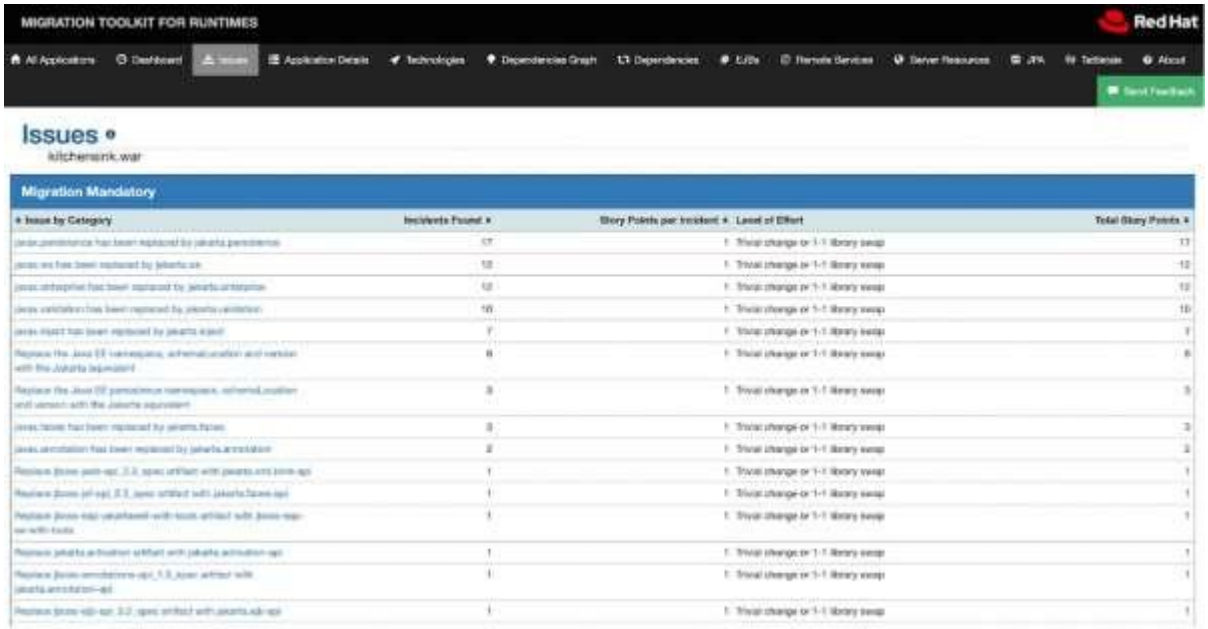

Figure 5: Migration Toolkit for Applications issues list.

The results from this analysis are in line with our expectations (i.e., changing “javax” references to “jakarta”). Some other changes are also

highlighted (e.g., our application uses Java Server Faces). So we need to update the version to match the JBoss EAP 8.0 version.

We can choose to either go through these items individually and make the code changes manually or use a tool like Openrewrite to automate these changes. For guidance on using Openrewrite to migrate from Java EE 8 to Jakarta EE 9, refer to this article.

One of the changes listed on the report is to update a dependency in the pom.xml file illustrated in Figure 6.

Figure 6: Migration Toolkit for Applications code change example.

In our example, in addition to changing the groupId and artifactId, we will need to change the version.server.bom property. In the pom.xml, replace the following:

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

with:

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

Once all these changes have been made, we should be able to build and deploy our application to JBoss EAP 8.0. To do this, with our JBoss EAP 8.0 server running, run the following command:

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

Once the maven process completes, our application should be available at http://127.0.0.1:8080/kitchensink.

Success! Our JBoss EAP 7.4 application and server configuration are now fully migrated to JBoss EAP 8.0.