


Deploy a Java application with Open Liberty/WebSphere Liberty on an Azure Red Hat OpenShift 4 cluster

10/30/2020 • 7 minutes to read •      +1

In this article

[Prerequisites](#)

[Prepare the Liberty application](#)

[Prepare the application image](#)

[Deploy application on the ARO 4 cluster](#)

This workshop demonstrates how to run your Java, Java EE, [Jakarta EE](#) , or [MicroProfile](#) application on the Open Liberty/WebSphere Liberty runtime and then deploy the containerized application to an Azure Red Hat OpenShift (ARO) 4 cluster using the Open Liberty Operator. This guide will walk you through preparing a Liberty application with [Azure PostgreSQL DB](#) , building the application Docker image and running the containerized application on an ARO 4 cluster. For more details on Open Liberty, see [the Open Liberty project page](#) . For more details on IBM WebSphere Liberty see [the WebSphere Liberty product page](#) .

Prerequisites

Complete the following prerequisites to successfully walk through this guide.

1. Clone the code for this sample on your local system. The sample is on [GitHub](#) .
2. Obtain your OpenShift console login credentials, the name of your OpenShift

- namespace, the OpenShift console URL, and the OpenShift container registry URL.
3. Obtain the Azure PostgreSQL server name, port, username and password.


Prepare the Liberty application

We'll use a Java EE 8 application as our example in this guide. Open Liberty is a [Java EE 8 full profile](#) compatible server, so it can easily run the application. Open Liberty is also [Jakarta EE 8 full profile compatible](#).

Run the application on Open Liberty

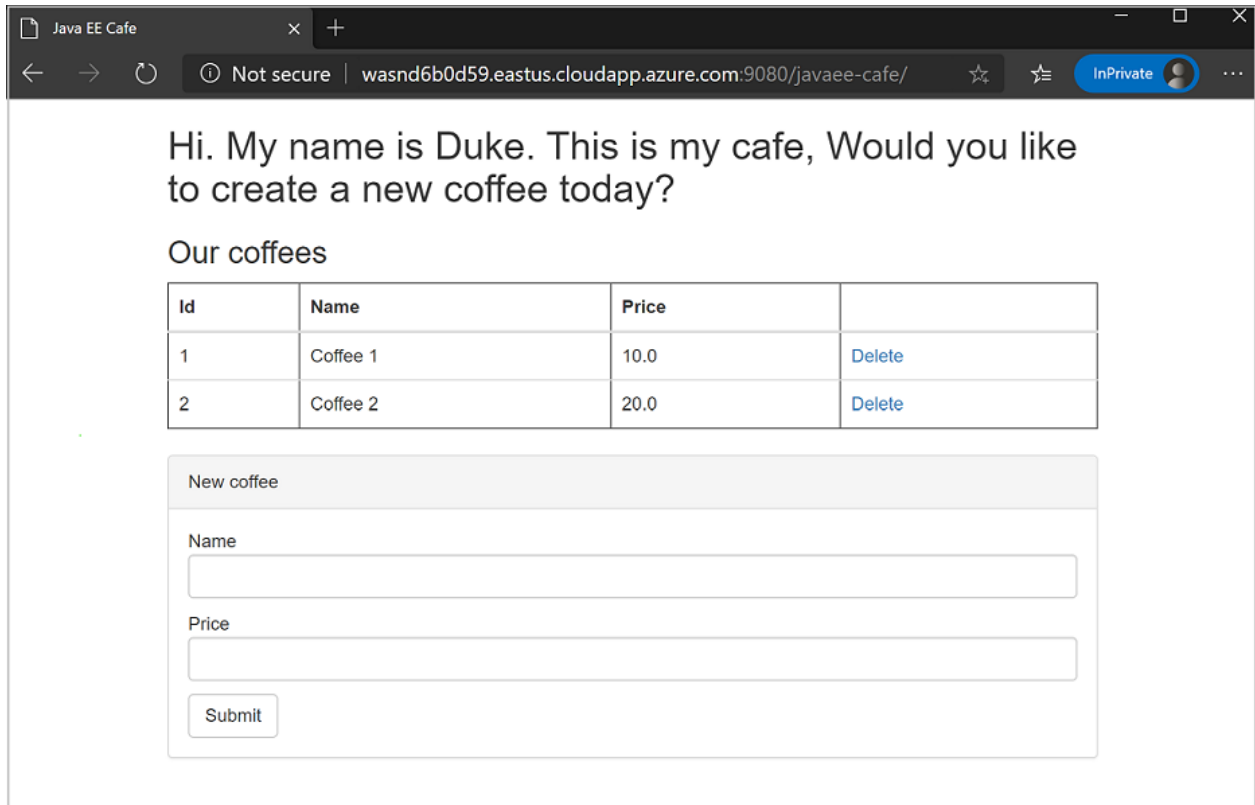
To run the application on Open Liberty, you need to create an Open Liberty server configuration file so that the [Liberty Maven plugin](#) can package the application for deployment. The Liberty Maven plugin is not required to deploy the application to OpenShift. However, we'll use it in this example with Open Liberty's developer (dev) mode. Developer mode lets you easily run the application locally. Complete the following steps on your local computer.

1. Change directory to `3-integration/connect-db/postgres` of your local clone.
2. Run `mvn clean package` in a console to generate a war package `javaee-cafe.war` in the directory `./target`.
3. Run `mvn -Ddb.server.name=<Server name>.postgres.database.azure.com -Ddb.port.number=<Port number> -Ddb.name=postgres -Ddb.user=<Admin username>@<Server name> -Ddb.password=<Password> liberty:dev` to start Open Liberty in dev mode.
4. Wait until the server starts. The console output should end with the following message:

| Text |  Copy |
|--|--|
| <pre>[INFO] CWWKM2015I: Match number: 1 is [6/10/20 10:26:09:517 CST] 00000022 com.ibm.ws.kernel.feature.internal.FeatureManager</pre> | |

```
A CWWKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 6.447 seconds..  
[INFO] Press the Enter key to run tests on demand. To stop the server and quit dev mode, use Ctrl-C or type 'q' and press the Enter key.  
[INFO] Source compilation was successful.
```

5. Open `http://localhost:9080/` in your browser to visit the application home page. The application will look similar to the following image:



6. Press **Control-C** to stop the application and Open Liberty server.


Prepare the application image

To deploy and run your Liberty application on an ARO 4 cluster, containerize your application as a Docker image using [Open Liberty container images](#) or [WebSphere Liberty container images](#).


Build application image

Complete the following steps to build the application image:

1. Change directory to <path-to-repo>/3-integration/connect-db/postgres of your local clone.
2. Download [postgresql-42.2.4.jar](#) and put it to current working directory.
3. Run the following commands to build application image.
 - Build with Open Liberty base image:

| | |
|--|--|
| Bash |  Copy |
| <pre># Build and tag application image. This will cause Docker to pull the necessary Open Liberty base images. [sudo] docker build -t javaee-cafe-connect-db-postgres:1.0.0 --pull --file=Dockerfile .</pre> | |

- Build with WebSphere Liberty base image:

| | |
|---|--|
| Bash |  Copy |
| <pre># Build and tag application image. This will cause Docker to pull the necessary WebSphere Liberty base images. [sudo] docker build -t javaee-cafe-connect-db-postgres:1.0.0 --pull --file=Dockerfile-wlp .</pre> | |

Run the application locally with Docker

Before deploying the containerized application to a remote cluster, run with your local Docker to verify whether it works:

1. Run `[sudo] docker run -it --rm -p 9080:9080 -e DB_SERVER_NAME=<Server name>.postgres.database.azure.com -e DB_PORT_NUMBER=<Port number> -e DB_NAME=postgres -e DB_USER=<Server admin login>@<Server name> -e`

DB_PASSWORD=<Password> javaee-cafe-connect-db-postgres:1.0.0 in your console.


2. Wait for Liberty server to start and the application to deploy successfully.
3. Open `http://localhost:9080/` in your browser to visit the application home page.
4. Press **Control-C** to stop the application and Liberty server.

Push the image to the container image registry

When you're satisfied with the state of the application, push it to the built-in container image registry by following the instructions below.

Log in to the OpenShift CLI with the provided credentials

1. Sign in to the OpenShift web console from your browser using the provided credentials.
 - a. Select **htpasswd**
2. Sign in with the OpenShift CLI by using the following steps. For discussion, this process is known as `oc login`.
 - a. At the right-top of the web console, expand the context menu of the signed-in user, then select **Copy Login Command**.
 - b. Sign in to a new tab window with the same user if necessary.
 - c. Select **Display Token**.
 - d. Copy the value listed below **Login with this token** to the clipboard and run it in a shell, as shown here.

| | |
|--|--|
| Bash |  Copy |
| <pre>oc login --token=X0dASlzeT7BHT0JZW6Fd4dl5EwHpeBlN27TAdWHseob -- server=https://api.aqlm62xm.rnfghf.aroapp.io:6443 Logged into "https://api.aqlm62xm.rnfghf.aroapp.io:6443" as</pre> | |

"**kube:admin**" using the token provided.

You have access to 57 projects, the list has been suppressed.
You can list all projects with '**oc projects**'

Using project "**default**".

Push the container image to the container registry for OpenShift

Execute these commands to push the image to the container registry for OpenShift.

Bash

 Copy

```
# Note: replace "<Container_Registry_URL>" with the fully qualified
name of the registry
Container_Registry_URL=<Container_Registry_URL>

# Create a new tag with registry info that refers to the source image
[sudo] docker tag javaee-cafe-connect-db-postgres:1.0.0 ${Contain-
er_Registry_URL}/<your-namespace>/javaee-cafe-connect-db-post-
gres:1.0.0

# Sign in to the built-in container image registry
[sudo] docker login -u $(oc whoami) -p $(oc whoami -t) ${Contain-
er_Registry_URL}
```

Successful output will look similar to the following.

Bash

 Copy

```
WARNING! Using --password via the CLI is insecure. Use --password-
stdin.
Login Succeeded
```

Push the image to the built-in container image registry with the following command.

Bash

 Copy

```
[sudo] docker push ${Container_Registry_URL}/<your namespace>/javaee-  
cafe-connect-db-postgres:1.0.0
```

Deploy application on the ARO 4 cluster

Now you can deploy the sample Liberty application to the Azure Red Hat OpenShift 4 cluster.

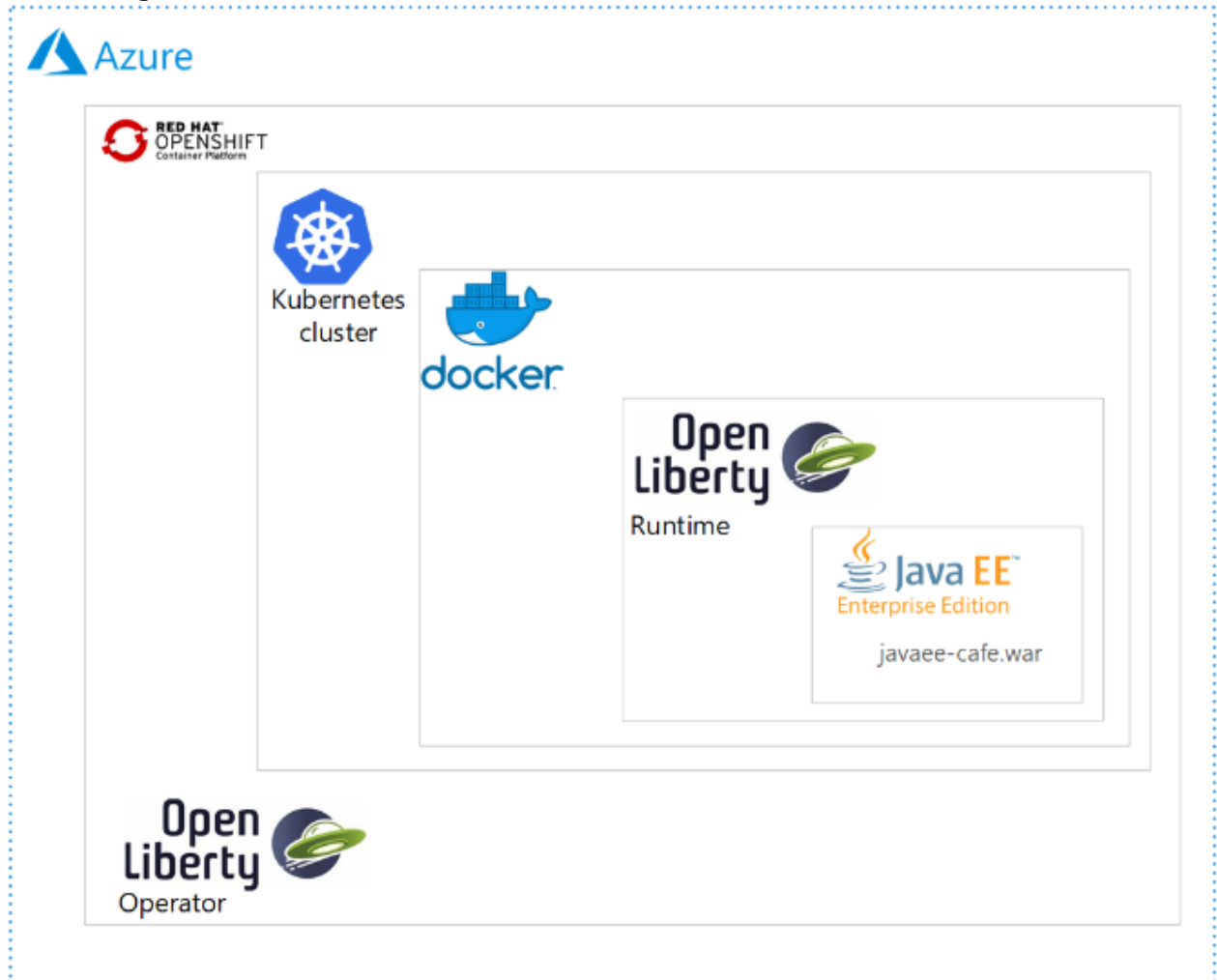
Deploy the application from the web console

Because we use the Open Liberty Operator to manage Liberty applications, we need to create an instance of its *Custom Resource Definition*, of type "OpenLibertyApplication". The Operator will then take care of all aspects of managing the OpenShift resources required for deployment.

1. Modify <path-to-repo>/3-integration/connect-db/openlibertyapplication.yaml to replace the following values:
 - Replace \${DB_Type} with postgres
 - Change the value of namespace: to be <Your namespace> instead of open-liberty-demo
 - Replace \${Image_Name} with javaee-cafe-connect-db-postgres
 - Replace \${DB_SERVER_NAME} with <Server name>.postgres.database.azure.com
 - Replace \${DB_PORT_NUMBER} with <Port number>
 - Replace \${DB_NAME} with postgres
 - Replace \${DB_USER} with <Admin username>@<Server name>
 - Replace \${DB_PASSWORD} with <Password>
2. Sign in to the OpenShift web console from your browser using the provided credentials.
3. In the top left section of the screen in the menu, ensure you have the

Administrator view instead of the Developer view. If you do not ensure this step you will not see the **Operators** option.

4. Expand **Home**, Select **Projects** > **your-namespace**.
5. Navigate to **Operators** > **Installed Operators**.
6. In the middle of the page, select **Open Liberty Operator**.
7. In the middle of the page, select **Open Liberty Application**. The navigation of items in the user interface mirrors the actual containment hierarchy of technologies in use.



8. Select **Create OpenLibertyApplication**
9. Replace the generated yaml with yours, which is located at `<path-to-repo>/3-integration/connect-db/openlibertyapplication.yaml`.
10. Select **Create**. You'll be returned to the list of OpenLibertyApplications.
11. Select **javaee-cafe-connect-db-postgres**.
12. In the middle of the page, select **Resources**.

13. In the table, select the link for **javaee-cafe-connect-db-postgres** with the **Kind of Route**.
14. On the page that opens, select the link below **Location**.

You'll see the application home page opened in the browser.

Delete the application from the web console


When you're done with the application, follow these steps to delete the application from Open Shift.

1. In the left navigation pane, expand the entry for **Operators**.
2. Select **Installed Operators**.
3. Select **Open Liberty Operator**.
4. In the middle of the page select **Open Liberty Application**.
5. Select the vertical ellipsis (three vertical dots) then select **Delete OpenLiberty Application**.

Deploy the application from CLI

Instead of using the web console GUI, you can deploy the application from the CLI.

1. Change directory to `3-integration/connect-db` of your local clone, and run the following commands to deploy your Liberty application to the ARO 4 cluster. Command output is also shown inline.

| | |
|--|--|
| Bash |  Copy |
| <pre># Switch to <your namespace> where resources of demo app will be- long to # Change directory to "<path-to-repo>/3-integration/connect-db" cd <path-to-repo>/3-integration/connect-db # Change project to your namespace oc project <your namespace> # Create OpenLibertyApplication "javaee-cafe-connect-db-postgres"</pre> | |

```
oc create -f openlibertyapplication.yaml

# Check if OpenLibertyApplication instance is created
oc get openlibertyapplication javaee-cafe-connect-db-postgres

# Check if deployment created by Operator is ready
oc get deployment javaee-cafe-connect-db-postgres
```

2. Check to see 2/2 under the READY column before you continue. If not, investigate and resolve the problem before continuing.
3. Discover the host of route to the application with the `oc get route` command, as shown here.

Bash

 Copy

```
# Get host of the route
HOST=$(oc get route javaee-cafe-connect-db-postgres --
template='{{ .spec.host }}')
echo "Route Host: $HOST"

Route Host: javaee-cafe-connect-db-
postgres.apps.aqlm62xm.rnfhf.aroapp.io
```

Once the Liberty application is up and running, open the output of **Route Host** in your browser to visit the application home page.

Delete the application from CLI


Delete the application from the CLI by executing this command.

Bash

 Copy

```
oc delete -f openlibertyapplication.yaml
```

Is this page helpful?

 Yes  No

Recommended content

[Tutorial - Create an Azure Red Hat OpenShift 4 cluster](#)

Learn how to create a Microsoft Azure Red Hat OpenShift cluster using the Azure CLI

[Introduction to Azure Red Hat OpenShift](#)

Learn the features and benefits of Microsoft Azure Red Hat OpenShift to deploy and manage container-based applications.

[Frequently asked questions for Azure Red Hat OpenShift](#)

Here are answers to common questions about Microsoft Azure Red Hat OpenShift

[Azure Red Hat OpenShift support lifecycle](#)

Understand the support lifecycle and supported versions for Azure Red Hat OpenShift

Show more 