Red Hat build of Quarkus 1.3

Deploying your Quarkus applications on Red Hat OpenShift Container Platform

Last Updated: 2020-12-08

Red Hat build of Quarkus 1.3 Deploying your Quarkus applications on Red Hat OpenShift Container Platform

Abstract

This guide describes how to deploy a Quarkus application on Red Hat OpenShift Container Platform.

Table of Contents

PREFACE

As an application developer, you can deploy a Quarkus application on Red Hat OpenShift Container Platform using Apache Maven and the **quarkus-openshift** extension in a single build command or you can use the traditional source-to-image (S2I) method. The resulting image from either method is fully supported. See the Development Support Scope of Coverage page for the build process and tools that are covered under development support.

Prerequisites

- OpenJDK 11 is installed and the **JAVA_HOME** environment variable specifies the location of the Java SDK. Red Hat build of Open JDK is available from the Software Downloads page in the Red Hat Customer Portal (login required).
- Apache Maven 3.6.2 or higher is installed. Maven is available from the Apache Maven Project website.
- You have a Quarkus Maven project. For instructions on building a simple Quarkus application with Maven, see *Getting started with Quarkus*.

  NOTE

  For a completed example of a Quarkus Maven project, download the Quarkus quickstart archive or clone the **Quarkus Quickstarts** Git repository. The example is in the **getting-started** directory.

- You have access to a Red Hat OpenShift Container Platform cluster and the latest version of the OpenShift CLI (oc) is installed. For information about installing oc, see the "Installing the CLI" section of the *Installing and configuring OpenShift Container Platform clusters* guide.

CHAPTER 1. RED HAT BUILD OF QUARKUS

Red Hat build of Quarkus is a Kubernetes-native Java stack that is optimized for use with containers and Red Hat OpenShift Container Platform. Quarkus is designed to work with popular Java standards, frameworks, and libraries such as Eclipse MicroProfile, Apache Kafka, RESTEasy (JAX-RS), Hibernate ORM (JPA), Spring, Infinispan, and Apache Camel.

The Quarkus dependency injection solution is based on CDI (contexts and dependency injection) and includes an extension framework to expand functionality and to configure, boot, and integrate a framework into your application.

Quarkus provides a container-first approach to building Java applications. This approach makes it much easier to build microservices-based applications written in Java as well as enabling those applications to invoke functions running on serverless computing frameworks. For this reason, Quarkus applications have small memory footprints and fast start-up times.

CHAPTER 2. RED HAT OPENSHIFT CONTAINER PLATFORM

Red Hat OpenShift Container Platform is a Kubernetes-based platform for developing and running containerized applications. It is designed to enable applications and the data centers that support them to expand from just a few systems and applications to thousands of systems that serve millions of clients.

OpenShift supports two workflows for building container images for applications: the source and the binary workflows. Both workflows are based on the source-to-image (S2I) feature and both workflows rely on S2I builds using the source workflow. The key difference is that the source workflow generates deployable artifacts of your application inside OpenShift, while the binary workflow generates these binary artifacts outside of OpenShift. Both of them build the application container image inside OpenShift. The binary workflow provides an application binary as the source for an OpenShift S2I build that generates a container image.

CHAPTER 3. USING THE QUARKUS OPENSHIFT EXTENSION TO DEPLOY QUARKUS APPLICATIONS ON OPENSHIFT

The traditional source-to-image (S2I) source workflow generates the deployable artifacts of your application inside OpenShift. The Quarkus OpenShift extension uses the S2I binary workflow to provide a more streamlined deployment process. Instead of building from the source, the extension uploads the JAR files from the local file system. As a result, the build process is up to ten times faster than the traditional S2I method. You can use the Quarkus OpenShift extension when developing locally as well as from a build server or continuous integration (CI) system to perform repeatable builds from source.

NOTE

Use the Quarkus OpenShift extension for development and testing purposes only. For production environments, consider using the traditional S2I method described in Chapter 4, *Using S2I to deploy Quarkus applications on OpenShift* .

Procedure

1. Change to the directory that contains your Quarkus Maven project.

2. To add the OpenShift extension to an existing project, enter the following command:

```
./mvnw quarkus:add-extension -Dextensions="openshift"
```

NOTE

You can include the **-Dextensions="openshift"** argument to add the Quarkus OpenShift extension when you create a new project.

When you add the OpenShift extension, the script adds the following dependency to the **pom.xml** file:

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-openshift</artifactId>
</dependency>
```

3. If you are using an untrusted certificate, add the following line to the **src/main/resources/application.properties** file:

```
quarkus.kubernetes-client.trust-certs=true
```

4. To direct OpenShift to use the Open JDK 11 Red Hat Enterprise Linux 7 image, add the following line to the **application.properties** file:

```
quarkus.s2i.base-jvm-image=registry.access.redhat.com/openjdk/openjdk-11-rhel7
```

NOTE

If you are deploying on IBM Z infrastructure, add **quarkus.s2i.base-jvm-image=registry.access.redhat.com/openj9/openj9-11-rhel8** to the **application.properties** file.

5. To create an OpenShift route, add the following line to the **application.properties** file:

```
quarkus.openshift.expose=true
```

6. Save the changes to the **application.properties** file.

7. Log in to the OpenShift CLI (oc):

```
oc login
```

8. To create a new OpenShift project, enter the following command where ***PROJECT_NAME*** is the name of your new project:

```
oc new-project PROJECT_NAME
```

9. To deploy your project to OpenShift, enter the following command in your Quarkus Maven project directory:

```
./mvnw clean package -Dquarkus.kubernetes.deploy=true
```

10. To display the names and routes of all deployed applications in the OpenShift project, enter the following command:

```
oc get route
```

11. To view the full URL to the application, where ***APPLICATION_NAME*** is the name of an application deployed in your OpenShift project, enter the following command:

```
export URL="http://$(oc get route APPLICATION_NAME -o jsonpath='{.spec.host}')"
echo "Application URL: $URL"
curl $URL/hello
```

12. To create an HTTP request on the route's **hello** endpoint, enter the following command:

```
curl $URL/hello
```

CHAPTER 4. USING S2I TO DEPLOY QUARKUS APPLICATIONS ON OPENSHIFT

The traditional source-to-image (S2I) method is widely used as the preferred method for deploying applications on Red Hat OpenShift Container Platform. With S2I, you must provide the source code to the build container either through a Git repository or by uploading the source at build time. Use this method to deploy your Quarkus applications in production environments.

Prerequisites

- You have a Quarkus Maven project hosted in a Git repository.

Procedure

1. Change to the directory that contains your Quarkus Maven project.
2. Create a hidden directory called **.s2i** at the same level as the **pom.xml** file.
3. Create a file called **environment** in the **.s2i** directory and add the following content:

   ```
   ARTIFACT_COPY_ARGS=-p -r lib/ *-runner.jar
   ```

4. Commit and push your changes to the remote Git repository.
5. Log in to the OpenShift CLI (oc):

   ```
   oc login
   ```

6. To create a new OpenShift project, enter the following command where ***PROJECT_NAME*** is the name of your new project:

   ```
   oc new-project PROJECT_NAME
   ```

7. To import the supported OpenShift image, enter the following command:

   ```
   oc import-image --confirm openjdk/openjdk-11-rhel7 --
   from=registry.access.redhat.com/openjdk/openjdk-11-rhel7
   ```

   NOTE

   If you are deploying on IBM Z infrastructure, enter **oc import-image --confirm openj9/openj9-11-rhel8 --from=registry.access.redhat.com/openj9/openj9-11-rhel8**.

   For information about this image, see the Red Hat OpenJDK 11 Java Applications page.

8. To build the project in OpenShift, enter the following command where ***GIT_PATH*** is the path to the Git repository that hosts your Quarkus project and ***PROJECT_NAME*** is the OpenShift project that you created.

   ```
   oc new-app openjdk-11-rhel7 GIT_PATH --name=PROJECT_NAME
   ```

NOTE

If you are deploying on IBM Z infrastructure, enter **oc new-app openj9/openj9-11-rhel8 *GIT_PATH* --name=*PROJECT_NAME***.

This command builds the project, creates the application, and deploys the OpenShift service.

9. To create an OpenShift route, enter the following command:

```
oc expose service/PROJECT_NAME
```

10. To view the new route, enter the following command where ***APPLICATION_NAME*** is the name of an application deployed in your OpenShift project:

```
export URL="http://$(oc get route APPLICATION_NAME -o jsonpath='{.spec.host}')"
echo "Application URL: $URL"
```

11. To create an HTTP request on the route's **hello** endpoint, enter the following command:

```
curl $URL/hello
```

12. To use your application, enter the URL returned in the preceding command in a web browser.

13. To deploy an updated version of the project, push any updates to the Git repository then enter the following command:

```
oc start-build PROJECT_NAME
```

14. Refresh your browser page after the build completes to see the changes.

Revised on 2020-12-08 20:07:33 UTC