Red Hat build of Quarkus 3.8

Deploying your Red Hat build of Quarkus applications to OpenShift Container Platform

Last Updated: 2024-07-24

Legal Notice

Abstract

This guide describes how to deploy Red Hat build of Quarkus applications to OpenShift Container Platform.

Table of Contents

PROVIDING FEEDBACK ON RED HAT BUILD OF QUARKUS DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to **create a ticket**.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. DEPLOYING YOUR RED HAT BUILD OF QUARKUS APPLICATIONS TO OPENSHIFT CONTAINER PLATFORM

As an application developer, you can deploy your Quarkus applications to Red Hat OpenShift Container Platform by using a single Maven command. This functionality is provided by the **quarkus-openshift** extension, which supports multiple deployment options, including the Docker build strategy and the Source-to-Image (S2I) strategy.

Here, you can learn about the preferred workflows to deploy your Quarkus applications to production environments. To learn about other ways to deploy Quarkus applications, see the Deploying on OpenShift guide in the Quarkus community.

Prerequisites

- You have OpenJDK 17 or 21 installed.
- You have set the **JAVA_HOME** environment variable to the location of the Java SDK.
- You have Apache Maven 3.8.6 or later installed.
- You have a Quarkus Maven project that includes the **quarkus-openshift** extension.
  - To add the Quarkus OpenShift extension, see Adding the Quarkus OpenShift extension .
- You have access to an OpenShift Container Platform cluster and the latest compatible version of the **oc** tool installed.
  - For information about installing the **oc** tool, see CLI tools.

1.1. OPENSHIFT CONTAINER PLATFORM BUILD STRATEGIES AND RED HAT BUILD OF QUARKUS

Red Hat OpenShift Container Platform is a Kubernetes-based platform for developing and running containerized applications. Although the Kubernetes upstream project provides additional strategies, Red Hat supports only the following strategies in Quarkus:

1.1.1. Overview of OpenShift Container Platform build strategies

Docker build

This strategy builds the artifacts outside the OpenShift Container Platform cluster, locally or in a CI environment, and provides them to the OpenShift Container Platform build system together with a Dockerfile. The artifacts include JAR files or a native executable. The container gets built inside the OpenShift Container Platform cluster and is provided as an image stream.

NOTE

The OpenShift Container Platform Docker build strategy is the preferred build strategy because it supports Quarkus applications targeted for JVM or compiled to native executables. However, for compatibility with earlier Quarkus versions, the default build strategy is S2I. To select the OpenShift Container Platform Docker build strategy, use the **quarkus.openshift.build-strategy** property.

Source to Image (S2I)

The build process is performed inside the OpenShift Container Platform cluster. Red Hat build of Quarkus fully supports using S2I to deploy Red Hat build of Quarkus as a JVM application.

Binary S2I

This strategy uses a JAR file as input to the S2I build process, which speeds up the building and deploying of your application.

1.1.2. Build strategies supported by Quarkus

The following table outlines the build strategies that Red Hat build of Quarkus 3.8 supports:

Build strategy	Support for Red Hat build of Quarkus tools	Support for JVM	Support for native	Support for JVM Serverless	Support for native Serverless
Docker build	YES	YES	YES	YES	YES
S2I Binary	YES	YES	NO	NO	NO
Source S2I	NO	YES	NO	NO	NO

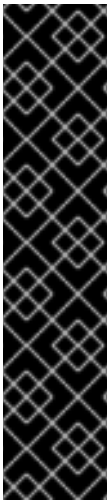
Additional resources

- Using S2I to deploy Quarkus applications to OpenShift Container Platform
- Deploying Quarkus Java applications to OpenShift Container Platform
- Deploying Quarkus applications compiled to native executables

1.2. ADDING THE RED HAT BUILD OF QUARKUS OPENSHIFT EXTENSION

To build and deploy your applications as a container image that runs inside your OpenShift Container Platform cluster, you must add the Red Hat build of Quarkus OpenShift extension **quarkus-openshift** as a dependency to your project.

The Quarkus OpenShift extension also generates OpenShift Container Platform resources such as image streams, build configuration, deployment, and service definitions. If your Quarkus application includes the **quarkus-smallrye-health** extension, OpenShift Container Platform can access the health endpoint and verify the startup, liveness, and readiness of your application.



IMPORTANT

From Red Hat build of Quarkus 3.8, the **DeploymentConfig** object, deprecated in OpenShift, is also deprecated in Red Hat build of Quarkus. **Deployment** is the default and preferred deployment kind for the Quarkus OpenShift extension.

If you redeploy applications that you deployed before by using **DeploymentConfig**, by default, those applications use **Deployment** but do not remove the previous **DeploymentConfig**.

This leads to a deployment of both new and old applications, so, you must remove the old **DeploymentConfig** manually. However, if you want to continue to use **DeploymentConfig**, it is still possible to do so by explicitly setting **quarkus.openshift.deployment-kind** to **DeploymentConfig**.

Prerequisites

- You have a Quarkus Maven project.
  - For information about how to create a Quarkus project with Maven, see Developing and compiling your Red Hat build of Quarkus applications with Apache Maven.

Procedure

1. To add the **quarkus-openshift** extension to your project, use one of the following methods:
   - Configure the **pom.xml** file:

     pom.xml

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

   - Enter the following command on the OpenShift Container Platform CLI:

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

   - Enter the following command on the Quarkus CLI:

     ```
     quarkus extension add 'quarkus-openshift'
     ```

1.3. SWITCHING TO THE REQUIRED OPENSHIFT CONTAINER PLATFORM PROJECT

You can use the Red Hat OpenShift Container Platform command-line interface (CLI) to create applications and manage your OpenShift Container Platform projects. Use the information provided to create an OpenShift Container Platform project or to switch to an existing one.

Prerequisites

- You have access to an OpenShift Container Platform cluster and the latest compatible version of the **oc** tool installed.

- For information about installing the **oc** tool, see CLI tools.

Procedure

1. Log in to the **oc** tool:

```
oc login
```

2. To show the current project space, enter the following command:

```
oc project -q
```

3. Use one of the following steps to go to the required OpenShift Container Platform project:

   a. If the project already exists, switch to the project:

```
oc project <project_name>
```

   b. If the project does not exist, create a new project:

```
oc new-project <project_name>
```

Additional resources

- Getting started with the OpenShift CLI

1.4. DEPLOYING RED HAT BUILD OF QUARKUS JAVA APPLICATIONS TO OPENSHIFT CONTAINER PLATFORM

The Red Hat build of Quarkus OpenShift extension enables you to deploy your Quarkus application to OpenShift Container Platform by using the Docker build strategy. The container gets built inside the OpenShift Container Platform cluster and is provided as an image stream.

Your Quarkus project includes pregenerated Dockerfiles with instructions. When you want to use a custom Dockerfile, you must add the file in the **src/main/docker** directory or anywhere inside the module. Additionally, you must set the path to your Dockerfile by using the **quarkus.openshift.jvm-dockerfile** property.

Prerequisites

- You have a Red Hat build of Quarkus Maven project that includes the **quarkus-openshift** extension.
- You are working in the correct OpenShift project namespace, as outlined in Switching to the required OpenShift Container Platform project.

Procedure

1. Set the Docker build strategy in your **application.properties** configuration file:

```
quarkus.openshift.build-strategy=docker
```

2. Optional: Set the following properties in the **application.properties** file, as required by your environment:

   a. If you are using an untrusted certificate, configure the **KubernetesClient**:

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

   b. Expose the service to create an OpenShift Container Platform route:

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

   c. Set the path to your custom Dockerfile:

   ```
   quarkus.openshift.jvm-dockerfile=<path_to_your_dockerfile>
   ```

   The following example shows the path to the **Dockerfile.custom-jvm**:

   ```
   quarkus.openshift.jvm-dockerfile=src/main/resources/Dockerfile.custom-jvm
   ```

3. Package and deploy your Quarkus application to the current OpenShift project:

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

Verification

The verification steps and related terminal outputs are demonstrated on the **openshift-helloworld** example application.

1. Display the list of pods associated with your current OpenShift project:

   ```
   oc get pods
   ```

   ```
   NAME                          READY   STATUS      RESTARTS   AGE
   openshift-helloworld-1-build    0/1     Completed   0          11m
   openshift-helloworld-1-deploy   0/1     Completed   0          10m
   openshift-helloworld-1-gzzrx    1/1     Running     0          10m
   ```

2. To retrieve the log output for your application's pod, use the **oc logs -f** command with the **<pod_name>** value of the pod you are interested in. In this example, we use the **openshift-helloworld-1-gzzrx** pod name that corresponds with the latest pod prefixed with the name of your application:

   ```
   oc logs -f openshift-helloworld-1-gzzrx
   ```

   ```
   Starting the Java application using /opt/jboss/container/java/run/run-java.sh ...
   INFO exec -a "java" java -Dquarkus.http.host=0.0.0.0 -
   Djava.util.logging.manager=org.jboss.logmanager.LogManager -
   XX:MaxRAMPercentage=50.0 -XX:+UseParallelGC -XX:MinHeapFreeRatio=10 -
   XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -
   XX:+ExitOnOutOfMemoryError -cp "." -jar /deployments/quarkus-run.jar
   __  ____  __  _____   ___  __ ____  ______
    --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
    -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
   ```

```
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2024-06-27 17:13:25,254 INFO  [io.quarkus] (main) getting-started 1.0.0-SNAPSHOT on
JVM (powered by Quarkus 3.8.5.redhat-00004) started in 0.653s. Listening on:
http://0.0.0.0:8080
2024-06-27 17:13:25,281 INFO  [io.quarkus] (main) Profile prod activated.
2024-06-27 17:13:25,281 INFO  [io.quarkus] (main) Installed features: [cdi, kubernetes,
resteasy-reactive, smallrye-context-propagation, vertx]
```

3. Retrieve a list of services:

```
oc get svc
```

```
NAME                  TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
openshift-helloworld   ClusterIP   172.30.64.57    <none>        80/TCP
14m
```

4. Get a URL to test your application.

NOTE

To create an OpenShift Container Platform route, ensure you have specified **quarkus.openshift.route.expose=true** in the **application.properties** file.

```
oc get routes
```

```
NAME                  HOST/PORT                                                PATH   SERVICES
PORT   TERMINATION   WILDCARD
openshift-helloworld   openshift-helloworld-username-dev.apps.sandbox-
m2.ll9k.p1.openshiftapps.com          openshift-helloworld   http              None
```

NOTE

Be aware that the route is now listening on port 80 and no longer at port 8080.

You can test the application demonstrated in this example with a web browser or a terminal by using **curl** and the complete URL output from **oc get routes**: http://openshift-helloworld-username-dev.apps.sandbox-m2.ll9k.p1.openshiftapps.com.

For example: **curl http://openshift-helloworld-username-dev.apps.sandbox-m2.ll9k.p1.openshiftapps.com**.

1.5. DEPLOYING RED HAT BUILD OF QUARKUS APPLICATIONS COMPILED TO NATIVE EXECUTABLES

You can deploy your native Red Hat build of Quarkus application to OpenShift Container Platform by using the Docker build strategy. You must create a native executable for your application that targets the Linux AMD64 operating system. If your host operating system is different from this, create a native Linux executable using a container runtime, for example, Docker or Podman.

Your Quarkus project includes pregenerated Dockerfiles with instructions. To use a custom Dockerfile, add the file in the **src/main/docker** directory or anywhere inside the module, and set the path to your Dockerfile by using the **quarkus.openshift.native-dockerfile** property.

Prerequisites

- You have a Linux AMD64 system or an Open Container Initiative (OCI) compatible container runtime, such as Podman or Docker.
- You have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- You are working in the correct OpenShift project namespace, as outlined in Switching to the required OpenShift Container Platform project.

Procedure

1. Set the Docker build strategy in your **application.properties** configuration file:

```
quarkus.openshift.build-strategy=docker
```

2. Set the container runtime:

```
quarkus.native.container-build=true
```

3. Optional: Set the following properties in the **application.properties** file, as required by your environment:
   a. If you are using an untrusted certificate, configure the **KubernetesClient** property:

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

   b. Expose the service to create an OpenShift Container Platform route:

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

   c. Set the path to your custom Dockerfile:

```
quarkus.openshift.native-dockerfile=<path_to_your_dockerfile>
```

   The following example shows the path to the **Dockerfile.custom-native**:

```
quarkus.openshift.jvm-dockerfile=src/main/docker/Dockerfile.custom-native
```

   d. Specify the container engine:
      - To build a native executable with Podman:

```
quarkus.native.container-runtime=podman
```

      - To build a native executable with Docker:

```
quarkus.native.container-runtime=docker
```

4. Finally, build a native executable, package, and deploy your application to OpenShift Container Platform:

```
./mvnw clean package -Pnative -Dquarkus.kubernetes.deploy=true (1)
```

(1) Compile the application to a native executable and enable Kubernetes deployment.

Verification

1. Verify that an image stream and a service resource is created and the Quarkus application is deployed by using the OpenShift web console. Alternatively, you can run the following OpenShift Container Platform command-line interface (CLI) commands:

```
oc get is (1)
oc get pods (2)
oc get svc (3)
```

(1) List the image streams created.

(2) View a list of pods associated with your current OpenShift project.

(3) Get the list of Kubernetes services.

2. To retrieve the log output for your application's pod, enter the following command where ***<pod_name>*** is the name of the latest pod prefixed with the name of your application:

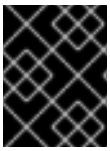
```
oc logs -f <pod_name>
```

Additional resources

- Managing image streams

1.6. USING S2I TO DEPLOY RED HAT BUILD OF QUARKUS APPLICATIONS TO OPENSHIFT CONTAINER PLATFORM

You can deploy your Red Hat build of Quarkus applications to OpenShift Container Platform by using the Source-to-Image (S2I) method. With S2I, you must provide the source code to the build container through a Git repository or by uploading the source at build time.



IMPORTANT

S2I is not supported for native deployments. For deploying Quarkus applications compiled to native executables, use the Docker build strategy.

The procedure for deploying your Quarkus applications to OpenShift Container Platform by using S2I differs depending on the Java version you are using.

1.6.1. Using S2I to deploy Red Hat build of Quarkus applications to OpenShift Container Platform with Java 17

You can deploy your Red Hat build of Quarkus applications running on Java 17 to OpenShift Container Platform by using the Source-to-Image (S2I) method.

Prerequisites

- You have a Quarkus application built with Java 17. For Java 17 applications, see Using S2I to deploy Red Hat build of Quarkus applications to OpenShift Container Platform with Java 17.
- (Optional): You have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- You are working in the correct OpenShift project namespace, as outlined in Switching to the required OpenShift Container Platform project.
- Your Quarkus Maven project is hosted in a Git repository.

Procedure

1. Open the **pom.xml** file, and change the Java configuration to version 17, as follows:

```
<maven.compiler.source>17</maven.compiler.source>
<maven.compiler.target>17</maven.compiler.target>
```

2. To package your Java 17 application, enter the following command:

```
./mvnw clean package
```

3. Create a directory called **.s2i** at the same level as the **pom.xml** file.
4. Create a file called **environment** in the **.s2i** directory and add the following content:

```
MAVEN_S2I_ARTIFACT_DIRS=target/quarkus-app
S2I_SOURCE_DEPLOYMENTS_FILTER=app lib quarkus quarkus-run.jar
JAVA_OPTIONS=-Dquarkus.http.host=0.0.0.0
AB_JOLOKIA_OFF=true
JAVA_APP_JAR=/deployments/quarkus-run.jar
```

5. Commit and push your changes to the remote Git repository.
6. To import the supported OpenShift Container Platform image, enter the following command:

```
oc import-image ubi8/openjdk-17 --from=registry.access.redhat.com/ubi8/openjdk-17 --
confirm
```

NOTE

- If you are using the OpenShift image registry and pulling from image streams in the same project, your pod service account should already have the correct permissions.
- If you are pulling images across other OpenShift Container Platform projects or from secured registries, additional configuration steps might be required.

For more information, see Using image pull secrets in Red Hat OpenShift Container Platform documentation.

7. To build the project, create the application, and deploy the OpenShift Container Platform service, enter the following command:

```
oc new-app registry.access.redhat.com/ubi8/openjdk-17~<git_path> --
name=<project_name>
```

   Where:

   - <git_path> is the path to the Git repository that hosts your Quarkus project. For example, **oc new-app registry.access.redhat.com/ubi8/openjdk-17~https://github.com/johndoe/code-with-quarkus.git --name=code-with-quarkus**. If you do not have SSH keys configured for the Git repository, when specifying the Git path, use the HTTPS URL instead of the SSH URL.
   - <project_name> is the name of your application.

8. 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>
```

9. To expose a route to the Quarkus application, enter the following command:

```
oc expose svc <project_name>
```

Verification

1. To view a list of pods associated with your current OpenShift project, enter the following command:

```
oc get pods
```

2. To retrieve the log output for your application's pod, enter the following command where **<pod_name>** is the name of the latest pod prefixed with the name of your application:

```
oc logs -f <pod_name>
```

Additional resources

- Red Hat build of OpenJDK applications in containers
- Route configuration

1.6.2. Using S2I to deploy Red Hat build of Quarkus applications to OpenShift Container Platform with Java 21

You can deploy your Red Hat build of Quarkus applications running on Java 21 to OpenShift Container Platform by using the Source-to-Image (S2I) method.

Prerequisites

- You have a Quarkus application built with Java 21. For Java 21 applications, see Using S2I to deploy Red Hat build of Quarkus applications to OpenShift Container Platform with Java 21.

- (Optional): You have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- You are working in the correct OpenShift Container Platform project namespace, as outlined in Switching to the required OpenShift Container Platform project .
- Your Quarkus Maven project is hosted in a Git repository.

Procedure

1. Open the **pom.xml** file, and change the Java configuration to version 21, as follows:

```
<maven.compiler.source>21</maven.compiler.source>
<maven.compiler.target>21</maven.compiler.target>
```

2. To package your Java 21 application, enter the following command:

```
./mvnw clean package
```

3. Create a directory called **.s2i** at the same level as the **pom.xml** file.
4. Create a file called **environment** in the **.s2i** directory and add the following content:

```
MAVEN_S2I_ARTIFACT_DIRS=target/quarkus-app
S2I_SOURCE_DEPLOYMENTS_FILTER=app lib quarkus quarkus-run.jar
JAVA_OPTIONS=-Dquarkus.http.host=0.0.0.0
AB_JOLOKIA_OFF=true
JAVA_APP_JAR=/deployments/quarkus-run.jar
```

5. Commit and push your changes to the remote Git repository.
6. To import the supported OpenShift Container Platform image, enter the following command:

```
oc import-image ubi8/openjdk-21 --from=registry.access.redhat.com/ubi8/openjdk-21 --
confirm
```

NOTE

- If you are using the OpenShift image registry and pulling from image streams in the same project, your pod service account should already have the correct permissions.
- If you are pulling images across other OpenShift Container Platform projects or from secured registries, additional configuration steps might be required. For more information, see Using image pull secrets in Red Hat OpenShift Container Platform documentation.
- If you are deploying on IBM Z infrastructure, enter **oc import-image ubi8/openjdk-21 --from=registry.redhat.io/ubi8/openjdk-21 --confirm** instead. For information about this image, see the Red Hat build of OpenJDK 21 page.

7. To build the project, create the application, and deploy the OpenShift Container Platform service, enter the following command:

```
oc new-app registry.access.redhat.com/ubi8/openjdk-21~<git_path> --
name=<project_name>
```

Where:

- <git_path> is the path to the Git repository that hosts your Quarkus project. For example, **oc new-app registry.access.redhat.com/ubi8/openjdk-21~https://github.com/johndoe/code-with-quarkus.git --name=code-with-quarkus**. If you do not have SSH keys configured for the Git repository, when specifying the Git path, use the HTTPS URL instead of the SSH URL.
- <project_name> is the name of your application.

NOTE

If you are deploying on IBM Z infrastructure, enter **oc new-app ubi8/openjdk-21~<git_path> --name=<project_name>** instead.

8. 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>
```

9. To expose a route to the Quarkus application, enter the following command:

```
oc expose svc <project_name>
```

Verification

1. To view a list of pods, enter the following command:

```
oc get pods
```

2. To retrieve the log output for your application's pod, enter the following command:

```
oc logs -f <pod_name>
```

Additional resources

- Red Hat build of OpenJDK applications in containers
- Route configuration

1.7. RED HAT BUILD OF QUARKUS CONFIGURATION PROPERTIES FOR CUSTOMIZING DEPLOYMENTS ON OPENSHIFT CONTAINER PLATFORM

You can customize your deployments on OpenShift Container Platform by defining optional configuration properties. You can configure your Red Hat build of Quarkus project in your **applications.properties** file or from the command line.

Table 1.1. Quarkus configuration properties and their default values:

Property	Description	Default
quarkus.container-image.group	The container image group. Must be set if the OpenShift Container Platform **<project_name>** is different from the username of the host system.	**${user.name}**
quarkus.container-image.registry	The container registry to use.	
quarkus.kubernetes-client.trust-certs	Kubernetes client certificate authentication.	
quarkus.kubernetes.deployment-target	Deployment target platform. For example, **openshift** or **knative**.	
quarkus.native.container-build	Builds a native Linux executable by using a container runtime. Docker is used by default.	**false**
quarkus.native.container-runtime	The container runtime used to build the image, for example, Docker.	
quarkus.openshift.build-strategy	The deployment strategy.	**s2i**
quarkus.openshift.route.expose	Exposes a route for the Quarkus application.	**false**
quarkus.native.debug.enabled	Enables debugging and generates debug symbols in a separate **.debug** file. When this property is used with **quarkus.native.container-build=true**, Red Hat build of Quarkus only supports Red Hat Enterprise Linux or other Linux distributions. The Red Hat Enterprise Linux and other Linux distributions contain the **binutils** package, which installs the **objcopy** utility to split the debug information from the native image.	**false**

1.8. ADDITIONAL RESOURCES

- Getting started with Quarkus
- OpenJDK Software Downloads

Revised on 2024-07-24 09:40:12 UTC