Red Hat build of Quarkus 1.11

Deploying your Quarkus applications to OpenShift

Last Updated: 2021-06-15

Red Hat build of Quarkus 1.11 Deploying your Quarkus applications to OpenShift

Legal Notice

Abstract

This guide describes how to deploy Quarkus applications to OpenShift.

Table of Contents

PREFACE

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

In the Red Hat build of Quarkus documentation you will learn the recommended workflows to deploy your Quarkus applications to production environments. To learn about alternative deployments, see the Quarkus community documentation.

Prerequisites

- Have OpenJDK (JDK) 11 installed and the **JAVA_HOME** environment variable set to specify the location of the Java SDK.
- Have Apache Maven 3.8.1 or higher installed.
- Have a Quarkus Maven project that includes the **quarkus-openshift** extension.
  - To add the Quarkus OpenShift extension, see Adding the Quarkus Openshift extension.
- Have access to a Red Hat OpenShift Container Platform cluster and the latest version of the OpenShift CLI (oc) installed.
  - For information about installing oc, see Installing and configuring OpenShift Container Platform clusters.
- Login to OpenShift using **oc** and select your project.
  - To verify the OpenShift project namespace, see Verifying the OpenShift project namespace.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our technical content and encourage you to tell us what you think. If you'd like to add comments, provide insights, correct a typo, or even ask a question, you can do so directly in the documentation.

NOTE

You must have a Red Hat account and be logged in to the customer portal.

To submit documentation feedback from the customer portal, do the following:

1. Select the **Multi-page HTML** format.
2. Click the **Feedback** button at the top-right of the document.
3. Highlight the section of text where you want to provide feedback.
4. Click the **Add Feedback** dialog next to your highlighted text.
5. Enter your feedback in the text box on the right of the page and then click **Submit**.

We automatically create a tracking issue each time you submit feedback. Open the link that is displayed after you click **Submit** and start watching the issue or add more comments.

Thank you for the valuable feedback.

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. OPENSHIFT BUILD STRATEGIES AND QUARKUS

Red Hat OpenShift Container Platform is a Kubernetes-based platform for developing and running containerized applications. For security and convenience OpenShift supports different build strategies that are not available in the upstream Kubernetes distributions.

Overview of OpenShift build strategies

Docker build

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

NOTE

Since Red Hat build of Quarkus 1.11, the OpenShift Docker build strategy is the preferred build strategy that supports Quarkus applications targeted for JVM as well as Quarkus applications compiled to native executables. However, the S2I remains the default deployment strategy for backwards compatibility reasons. You can configure the deployment strategy using the **quarkus.openshift.build-strategy** property.

Source to Image (S2I)

The build process is performed inside the OpenShift cluster. Using S2I to deploy Quarkus as a JVM application is fully supported.

Binary S2I

This strategy uses a JAR file as an input to the S2I build process. This speeds up the build process and deployment of your application.

Table 1.1. Quarkus support OpenShift build strategies

Build strategy	Support for Quarkus tooling	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
- Deploying Quarkus Java applications to OpenShift
- Deploying Quarkus applications compiled to native executables

CHAPTER 2. ADDING THE QUARKUS OPENSHIFT EXTENSION

You need to add the **quarkus-openshift** extension as a dependency to your Quarkus project so that you can build and deploy your applications as a container image to be used inside your OpenShift cluster.

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

Prerequisites

- Have a Quarkus Maven project.
  - For information on how to create Quarkus project with Maven, see Developing and compiling your Quarkus applications with Apache Maven.

Procedure

1. Change to the directory that contains your Quarkus project.
2. Use one of the following methods to add the **quarkus-openshift** extension to your project:
   a. Add the **quarkus-openshift** extension to the **pom.xml** file:

   pom.xml

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

   b. Add the **quarkus-openshift** extension using the command line:

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

CHAPTER 3. DEPLOYING QUARKUS JAVA APPLICATIONS TO OPENSHIFT

By using the Quarkus OpenShift extension, you can deploy your application to OpenShift using the Docker build strategy. The container is built inside the OpenShift cluster and provided as an image stream.

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

Prerequisites

- Have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- Login to OpenShift using **oc** and select your project.
  - To verify the OpenShift project namespace, see Verifying the OpenShift project namespace.

Procedure

1. Change to the directory that contains your Quarkus project.
2. Configure the following properties in your **application.properties** file:
   a. Set the Docker build strategy:

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

   b. (Optional) If you are using an untrusted certificate, configure the **KubernetesClient**:

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

   c. (Optional) Expose the service to create an OpenShift route:

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

   d. (Optional) 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

1. View a list of pods associated with your current OpenShift project:

   ```
   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>
   ```

CHAPTER 4. DEPLOYING QUARKUS APPLICATIONS COMPILED TO NATIVE EXECUTABLES

You can deploy your native Quarkus application to OpenShift using the Docker build strategy. You need to create a native executable for your application that targets the Linux X86_64 operating system. If your host operating system is different from this, you will need to create a native Linux executable using a container runtime like Docker or Podman.

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

Prerequisites

- A Linux X86_64 operating system or an OCI (Open Container Initiative) compatible container runtime, such as Podman or Docker.
- Have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- Login to OpenShift using **oc** and select your project.
  - To verify the OpenShift project namespace, see Verifying the OpenShift project namespace.

Procedure

1. Change to the directory that contains your Quarkus project.
2. Configure the following properties in your **application.properties** file:
   a. Set the Docker build strategy:

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

   b. Set the container runtime:

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

   c. (Optional) If you are using an untrusted certificate, configure the **KubernetesClient**:

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

   d. (Optional) Expose the service to create an OpenShift route:

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

   e. (Optional) 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
```

f. (Optional) 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
  ```

3. Build a native executable, package, and deploy your application to OpenShift:

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

Verification

1. View a list of pods associated with your current OpenShift project:

   ```
   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>
   ```

CHAPTER 5. DEPLOYING QUARKUS APPLICATIONS AS AN OPENSHIFT SERVERLESS SERVICE

You can deploy your Quarkus applications to OpenShift Serverless using the Docker build strategy. By using OpenShift Serverless Knative Serving, you can scale services up and down depending on the load size. Scaling down services that are currently not requested improves memory capabilities.

Your Quarkus project includes pre-generated Dockerfiles with instructions. When you want to use a custom Dockerfile, you need to add the file in the **src/main/docker** directory or anywhere inside the module. Additionally, you need to set the path to your Dockerfile using the **quarkus.openshift.jvm-dockerfile** property for JVM mode and **quarkus.openshift.native-dockerfile** property for native mode.

The following procedure demonstrates how to deploy a Serverless Quarkus Java application or a Serverless application compiled to a native executable using the Quarkus OpenShift extension.

Prerequisites

- Have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- Login to OpenShift using **oc** and select your project.
  - To verify the OpenShift project namespace, see Verifying the OpenShift project namespace.
- OpenShift Serverless operator is installed.
- OpenShift Knative Serving is installed and verified. See Installing Knative Serving.
- For native compilation, a Linux X86_64 operating system or an OCI (Open Container Initiative) compatible container runtime, such as Podman or Docker is required.

Procedure

1. Change to the directory that contains your Quarkus project.
2. Configure the following properties in your **application.properties** file:
   a. Set Knative as a deployment target:

      ```
      quarkus.kubernetes.deployment-target=knative
      ```

   b. Set the Docker build strategy:

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

   c. Direct OpenShift Serverless to pull your container image from the OpenShift internal registry:

      ```
      quarkus.container-image.registry=image-registry.openshift-image-registry.svc:5000
      ```

> **NOTE**
>
> If your OpenShift **<project_name>** is different from the **username** of the host system, set the group for the container image otherwise Quarkus cannot pull the image from the image registry.
>
> ```
> quarkus.container-image.group=<project_name>
> ```

d. (Optional) If you are using an untrusted certificate, configure the **KubernetesClient**:

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

e. (Optional) Expose the service to create an OpenShift route:

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

f. (Optional) 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. (Optional) To deploy a Serverless application compiled to a native executable, you need to configure the following properties:

a. Set the container runtime:

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

b. 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
```

c. (Optional) Set the path to your custom Dockerfile:

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

4. Package and deploy your Serverless application to OpenShift using one of the following options:

a. Deploy a Quarkus Java application:

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

b. Deploy a Quarkus native application:

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

Verification

1. View a list of pods associated with your current OpenShift project:

```
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>
```

CHAPTER 6. USING S2I TO DEPLOY QUARKUS APPLICATIONS TO OPENSHIFT

You can deploy your Quarkus applications to OpenShift using the Source-to-Image (S2I) method. 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.

IMPORTANT

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

Prerequisites

- Have a Quarkus Maven project that includes the **quarkus-openshift** extension.
- Host your Quarkus Maven project in a Git repository.
- Login to OpenShift using **oc** and select your project.
  - To verify the OpenShift project namespace, see Verifying the OpenShift project namespace.

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

```
oc import-image --confirm ubi8/openjdk-11 --from=registry.access.redhat.com/ubi8/openjdk-11
```

NOTE

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

For information about this image, see the Red Hat OpenJ9 11 Java Applications on RHEL8 page.

6. 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 ubi8/openjdk-11 <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.

7. 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>
```

Verification

1. View a list of pods associated with your current OpenShift project:

```
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

- OpenJDK applications in containers

CHAPTER 7. VERIFYING THE OPENSHIFT PROJECT NAMESPACE

Prerequisites

- Have access to a Red Hat OpenShift Container Platform cluster and the latest version of the OpenShift CLI (oc) installed.

Procedure

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

   ```
   oc login
   ```

2. Display the current project space:

   ```
   oc project -q
   ```

3. (Optional) Create a new OpenShift project:

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

Additional resources

- Getting started with the OpenShift CLI

CHAPTER 8. QUARKUS CONFIGURATION PROPERTIES FOR CUSTOMIZING DEPLOYMENTS ON OPENSHIFT

You can customize your deployments on OpenShift by defining optional configuration properties. You can configure your Quarkus project in your **applications.properties** file or via the command line.

Table 8.1. Table Quarkus configuration properties and their default values:

Property	Description	Default
quarkus.container-image.group	The container image group. Must be set if the OpenShift **<project_name>** is different from a 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 using a container runtime. Docker is used by default	**false**
quarkus.native.container-runtime	The container runtime used build the image. For example, Docker	
quarkus.openshift.build-strategy	The deployment strategy	**s2i**
quarkus.openshift.expose	Exposes a route for the Quarkus application	**false**
quarkus.native.debug.enabled	Enables debug and generates debug symbols in a separate .debug file. When used with **quarkus.native.container-build=true**, Red Hat build of Quarkus only supports Red Hat Enterprise Linux or other Linux distributions as they contain the **binutils** package that installs the **objcopy** utility to split the debug info from the native image.	**false**

CHAPTER 9. ADDITIONAL RESOURCES

- Developing and compiling your Quarkus applications with Apache Maven
- Compiling your Quarkus applications to native executables
- Learning how to use the command-line tools for OpenShift Container Platform
- Installing and configuring OpenShift Container Platform clusters

Revised on 2021-06-15 14:51:41 UTC